

**In the United States Patent and Trademark Office
on Appeal from the Examiner to the Board
of Patent Appeals and Interferences**

In re Application of: Shmuel Shaffer
Serial No.: 10/824,180
Filing Date: April 14, 2004
Group Art Unit: 2614
Confirmation No.: 6361
Examiner: Khai N. Nguyen
Title: *Enhanced Extension Mobility*

Mail Stop: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Appeal Brief

Appellants have appealed to the Board of Patent Appeals and Interferences ("Board") from the decision of the Examiner mailed April 29, 2010, finally rejecting Claims 1-45. Appellants filed a Notice of Appeal on July 29, 2010, along with a Pre-Appeal Brief Request for Review. A Notice of Panel Decision from Pre-Appeal Brief Request for Review was mailed September 27, 2010, indicating that the case should proceed to the Board. A previous Appeal Brief was filed in this case on October 6, 2008, and the then-statutory fee of \$510.00 was paid with the filing of that previous Appeal Brief. Since no final Board decision was reached after the filing of the previous Appeal Brief, only the \$30.00 difference between the current statutory fee of \$540.00 and the previously-paid \$510.00 fee is believed to be due for this Appeal Brief. *See* M.P.E.P. § 1204.01.

Real Party In Interest

This Application is currently owned by Cisco Technology, Inc. as indicated by an assignment recorded on April 14, 2004, in the Assignment Records of the United States Patent and Trademark Office (PTO) at Reel 015243, Frame 0325 (5 pages).

Related Appeals and Interferences

To the knowledge of Appellants' counsel, there are no appeals, interferences, or judicial proceedings that are related to or will directly affect, be directly affected by, or have a bearing on the Board's decision regarding this Appeal.

Status of Claims

Claims 1-45 are pending in this Application¹ and stand rejected pursuant to the Final Office Action mailed April 29, 2010 (the "Final Office Action") for the following reasons:

- Claims 1-8, 10-20, 22-34, and 36-42 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication No. 2005/0180555 by Sarp et al. ("*Sarp*");
- Claims 9, 21, and 35 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Sarp* in view U.S. Patent 5,933,488 to Marcus et al. ("*Marcus*"); and
- Claims 43-45 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Sarp* in view U.S. Patent 6,292,792 to Baffes et al. ("*Baffes*")

Claims 1-45 are presented for Appeal. All pending claims are shown in Appendix A.

¹ In the Final Office Action, the Examiner objected to Claims 7, 19, 33, and 42 for including a typographical error. To simplify issues for Appeal, Appellants submitted an Amendment pursuant to 37 C.F.R. §41.33(a) amending Claims 7, 19, 33, and 42 to correct the identified typographical error. In this Appeal Brief, Appellants treat these amendments as having been entered.

Status of Amendments

On October 22, 2010, Appellants submitted an Amendment pursuant to 37 C.F.R. §41.33(a) amending Claims 7, 19, 33, and 42 to correct typographical errors identified by the Examiner in the Final Office Action. These amendments simplify issues for Appeal and do not affect the scope of any other pending claim on Appeal. Accordingly, Appellants treat this Amendment as having been entered for purposes of this Appeal. All other amendments submitted by Appellants were entered by the Examiner prior to the mailing of the Final Office Action.

Summary of Claimed Subject Matter

In certain embodiments, the present invention is operable to provide enhanced extension mobility (EEM) in a communication network. FIGURE 1 illustrates one embodiment of a communication system 10 operable to provide EEM in a communication network 12. Communication networks 12a and 12b are local area networks (LANs). Communication network 12c is a public switched telephone network (PSTN). Communication networks 12 are coupled to each other using network links 14 that may each include one or more LANs, wide area networks (WANs), metropolitan area networks (MANs), portions of the Internet, PSTNs, or other network links 14 or a combination of two or more such network links 14. In particular embodiments, a contact admission control (CAC) system is used to monitor bandwidth availability over a WAN coupling two or more communication networks 12 to each other.

One or more portions of a communication network 12 may be associated with a particular enterprise or other organization. Another organization may operate one or more such portions of communication network 12 according to an outsourcing arrangement between the two organizations. In addition, one or more of portions of communication network 12 may include a private communication network 12, a virtual communication network 12, or both. One or more portions of communication network 12 may include one or more trust domains. One or more of portions of communication network 12 may be a distributed communication network 12.

A communication network 12 may include one or more network devices. A network device includes one or more hardware components, software components, or embedded-logic components or a combination of two or more such components supporting communication among multiple endpoints 16. As an example and not by way of limitation, a network device may include one or more network components, gatekeepers, contact managers, routers, hubs, switches, gateways, or endpoints 16 or a combination of two or more such devices. In particular embodiments, a network device may be an automatic contact distributor (ACD) coupled to one or more endpoints 16. An ACD includes a specialized communication system for routing incoming contacts to available agents at endpoints 16 coupled to the ACD. The ACD may route incoming contacts so that they are properly distributed among available agents. A contact includes a request for service communicated using any audio and/or video means, including signals, data or messages transmitted through voice devices, text chat, web sessions, facsimile, instant messaging and e-mail. Network devices in a communication

network 12 may be coupled to each other according to any suitable arrangement using one or more network segments. A network segment may include one or more communication networks 12, computer buses, wireline segments, optical segments, wireless segments, or other segments or a combination of two or more of such segments.

Communication networks 12 each have endpoints 16. An endpoint 16 includes one or more hardware components, software components, or embedded-logic components or a combination of two or more such components for communicating with one or more other endpoints 16. As an example and not by way of limitation, an endpoint 16 may include a phone (which may be a mobile phone, a desktop phone, or another phone), a computer (which may be a laptop computer, a desktop computer, or other computer), a personal digital assistant (PDA), a video monitor, a camera, a fax machine, or other device. An endpoint 16 may be coupled to a network device in a communication network 12 using one or more endpoint links 18 that may each include one or more computer buses, LANs, MANs, WANs, or portions of the Internet or any other appropriate wireline, optical, wireless, or other endpoint links 18. Endpoints 16 may communicate with each other using packets of data. A packet may include one or more packets, cells, frames, or other units of data. Data may include one or more data components, metadata components, executable software components, or other components.

Endpoints 16 may use one or more suitable communication protocols to communicate with each other. According to one or more such communication protocols, one or more endpoints 16 may each be identified using a unique address. In addition or as an alternative, one or more network devices may each be identified using a unique address. As an example and not by way of limitation, in particular embodiments, two or more endpoints 16 may each be identified by an Internet Protocol (IP) address and may communicate with each other using IP. In these embodiments, one or more components of system 10 may support point-to-point, multicast, unicast, or other communication. One or more endpoints 16 and network devices may support Voice over IP (VoIP) or Voice over Packet (VoP). To communicate using VoIP or VoP, an endpoint packetizes voice data into packets communicable over one or more packet-based communication networks 12. Endpoints 16 and network devices that may support VoIP or VoP include telephones, fax machines, computers running telephony software, nodes, gateways, and other devices capable of providing telephony functionality over a packet-based communication network 12.

Communication between a first endpoint 16 and one or more second endpoints 16 may include one or more voice components, text components, executable software components, data components, or other components or a combination of two or more such components. As an example and not by way of limitation, a communication between a first endpoint 16 and one or more second endpoints 16 may include one or more instant messages (IMs). One or more endpoints 16 and network devices may support use of Session Initiation Protocol (SIP) for IM and possibly other functionality. In addition or as an alternative, one or more endpoints 16 and network devices may support use of SIP for Instant Messaging and Presence-Leveraging Extensions (SIMPLE) Protocol. In system 10, one or more voice-enabled endpoints 16 may support use of SIP and presence-related applications. In addition or as a further alternative, one or more endpoints 16 and network devices may support use of Instant Messaging and Presence Protocol (IMPP). Reference to "IM" may encompass both IM and one or more IM-related protocols.

A communication network 12 may receive incoming phone calls from first endpoints 16 and route the incoming phone calls to second endpoints 16 coupled to communication network 12. Communication network 12 may route incoming phone calls according to calling phone numbers, called phone numbers, or both. An endpoint 16 may support an extension of a user. In particular embodiments, an extension includes a number or a series of numbers corresponding to a first user. One or more second users may use the extension to contact the first user. Communication network 12 may provide extension mobility or similar functionality enabling "hotelling" or a similar feature. Such functionality may allow a user to logon at a first endpoint 16 so that first endpoint 16 supports an extension of the first user and then log off at first endpoint 16. The user may then logon at a second endpoint 16 so that second endpoint 16 supports the extension of the user, which may provide mobility to the user in communication network 12. In particular embodiments, a user may be concurrently logged on at multiple endpoints 16.

Communication network 12a includes an EEM server 20 and EEM data 22. EEM server 20 may interact with EEM clients 24 at endpoints 16 coupled to communication network 12a to provide EEM functionality to endpoints 16, as described below. EEM server 20 and EEM clients 24 may use EEM data 22 to provide such functionality, as described below. Although EEM server 20 is illustrated and described as providing EEM functionality to endpoints 16 coupled to communication network 12a, the present invention contemplates server 20 providing EEM functionality to any suitable endpoints 16 coupled to any suitable

communication networks 12. EEM client 24 may be either a thin (browser based) or a thick client.

An EEM client 24 at an endpoint 16 includes one or more hardware components, software components, or embedded logic components or a combination of two or more such components providing EEM functionality at endpoint 16, as described below. An endpoint 16 including an EEM client 24 may have a private mode and a shared mode. In private mode, endpoint 16 supports only one extension. In shared mode, endpoint 16 concurrently supports multiple extensions. An endpoint 16 may receive an incoming phone call that has a called extension corresponding to an extension supported at endpoint 16 and, in response to receiving the incoming phone call, prompt a called user of the incoming phone call to answer the incoming phone call. In particular embodiments, in private mode, endpoint 16 may receive the incoming phone call only if the called extension corresponds to the one extension supported at endpoint 16. Because endpoint 16 supports only one extension in private mode, endpoint 16 need not (but may nonetheless) identify the called extension when endpoint 16 prompts the called user to answer the incoming phone call. In particular embodiments, in shared mode, endpoint 16 may receive the incoming phone call if the called extension corresponds to any one of the multiple extensions supported at endpoint 16.

Because endpoint 16 supports multiple extensions in shared mode, endpoint 16 may identify the called extension when endpoint 16 prompts the called user to answer the incoming phone call. As an example and not by way of limitation, endpoint 16 may display the called extension, a name of the called user, or both at a display screen of endpoint 16. As another example, endpoint 16 may audibly announce a name of the called user between rings. In particular embodiments, endpoint 16 may access a .wav or similar file to audibly announce a name of a user. The .wav or similar file may be stored in a field (such as a UserName field) in a directory in EEM data 22 and downloaded to endpoint 16 when the user logs on at endpoint 16. As another example, endpoint 16 may play a unique ring tone that identifies the called user. In particular embodiments, when a user logs on at endpoint 16, endpoint 16 may present a unique ring tone to the user so that the user will later be able to identify incoming phone calls for the user. In particular embodiments, the user may select the unique ring tone at logon.

When an outgoing phone call is placed from an endpoint 16, endpoint 16 may generate signaling data identifying a calling extension of the outgoing phone call, a calling user of the outgoing phone call, or both for communication with the outgoing phone call. In

particular embodiments, in private mode, endpoint 16 may generate the signaling data according to the one extension supported at endpoint 16. In particular embodiments, in shared mode, endpoint 16 may generate the signaling data according to user input specifically identifying the calling extension. As an example and not by way of limitation, when the outgoing phone call is placed, the calling user may press a button at endpoint 16 specifically identifying the calling user. As another example, when the outgoing phone call is placed, endpoint 16 may prompt the calling user to identify the calling user. To prompt the calling user to identify the calling user, endpoint 16 may audibly announce the following menu options: "Press 1 if you are Joe. Press 2 if you are Mark." The calling user may then select the menu option corresponding to the calling user to identify the calling user. In particular embodiments, when endpoint 16 receives user input identifying the calling user, endpoint 16 may present a dial tone for placing the outgoing phone call. In particular embodiments, in private mode, endpoint 16 may generate the signaling data according to a predetermined extension. As an example and not by way of limitation, endpoint 16 may be configured so that, in shared mode, endpoint 16 generates signaling data for every outgoing phone call placed at endpoint 16 according to an extension of an owner of endpoint 16. Endpoint 16 may be located in an office of the owner of endpoint 16.

An endpoint 16 may assume one or more preferences of a user whose extension is supported at endpoint 16. To assume a preference of a user, endpoint 16 may download the preference from a profile of the user in EEM data 22 when the user logs on at endpoint 16. In particular embodiments, in private mode, endpoint 16 may assume all preferences of the one user whose extension is supported at endpoint 16. In particular embodiments, in shared mode, endpoint 16 may assume one or more preferences of one or more of the multiple users whose extensions are supported at endpoint 16. When endpoint 16 receives an incoming phone call that has a called extension corresponding to an extension of a user supported at endpoint 16, endpoint 16 may exhibit one or more preferences of that user with respect to the incoming phone call. Similarly, when a user whose extension is supported at endpoint 16 places an outgoing phone call, endpoint 16 may exhibit one or more preferences of that user with respect to the outgoing phone call. In particular embodiments, in shared mode, endpoint 16 may assume one or more preferences of only one user whose extension is supported at endpoint 16. Endpoint 16 may be configured so that, in shared mode, endpoint 16 assumes all preferences of an owner of endpoint 16 and no preferences of any other user. In particular embodiments, when a user whose extension is supported at endpoint 16 places an outgoing

phone call, call detail records (CDRs) and billing information may be updated to indicate that the user made the outgoing phone call. The CDRs and billing information may be stored at EEM data 22 or elsewhere, according to particular needs.

To access EEM functionality at an endpoint 16, a user may invoke an EEM client 24 at endpoint 16. When the user invokes EEM client 24, EEM client 24 may cause endpoint 16 to provide a logon menu to the user. In particular embodiments, endpoint 16 displays the logon menu at a display screen of endpoint 16. The logon menu may prompt the user to identify the user, provide a valid password, or both to logon at endpoint 16. In particular embodiments, the logon menu prompts the user to enter an extension of the user and a password corresponding to that extension. If the user enters a valid extension and a valid password corresponding to that extension, EEM client 24 may cause endpoint 16 to provide the following menu option to the user: "Enter 1 to logon in private mode. Enter 2 to logon in shared mode." If the user selects private mode, EEM client 24 may configure endpoint 16 to support only the extension entered by the user, as described above. If the user selects shared mode, EEM client 24 may configure endpoint 16 to support the extension entered by the user in addition to one or more other extensions already supported at endpoint 16, as described above. To configure endpoint 16 to support the extension entered by the user, EEM client 24 may establish one or more new lines or other connections at endpoint 16 for the extension entered by the user. EEM client 24 may communicate with EEM server 20 to establish the one or more new lines or other connections at endpoint 16. In addition, EEM client 24 may communicate with EEM server 20 to access EEM data 22, as described above.

FIGURE 2 illustrates an example method for EEM. The method begins at step 100, where a user invokes an EEM client 24 at an endpoint 16. At step 102, endpoint 16 prompts the user to enter an extension and a password. At step 104, the user enters an extension and a password. At step 106, endpoint 16 determines whether the extension and the password are valid. If the extension or the password is invalid, the method returns to step 102. If the extension and the password are valid, the method proceeds to step 108, where endpoint 16 prompts the user to select private mode or shared mode. At step 110, the user selects private mode or shared mode. At step 112, endpoint 16 determines whether the user selected private mode. If the user selected private mode, the method proceeds to step 114. At step 114, EEM client 24 configures endpoint 16 to support only the extension entered by the user at step 104, at which point the method ends. Returning to step 112, if the user did not select private mode, the method proceeds to step 116. At step 116, endpoint 16 determines whether the

user selected shared mode. If the user selected shared mode, the method proceeds to step 118, where EEM client 24 configures endpoint 16 to support the extension entered by the user at step 104 in addition to one or more other extensions of one or more other users already supported at endpoint 16, at which point the method ends. Returning to step 116, if the user did not select shared mode, the method returns to step 108.

With regard to the independent claims currently under Appeal, Appellants provide the following concise explanation of the subject matter recited in the claim elements. For brevity, Appellants do not necessarily identify every portion of the Specification and drawings relevant to the recited claim elements. Additionally, this explanation should not be used to limit Appellants' claims but is intended to assist the Board in considering the Appeal of this Application.

For example, independent Claim 1 recites the following:

A system for enhanced extension mobility (*see, e.g.*, Fig. 1; Spec. at 3:4; and 6:2-10:22), the system comprising one or more processing units collectively operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 3:5-11; 9:19-11:7); and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 11:23-14:2; 14:8-29).

As another example, independent Claim 15 recites the following:

A method for enhanced extension mobility (*see, e.g.*, Fig. 2; Spec. at 3:4; and 6:2-10:22), the method comprising:

accessing user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 3:5-11; 9:19-11:7); and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configuring the endpoint to concurrently support an extension of the user and one or more other

extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 11:23-14:2; 14:8-29).

As another example, independent Claim 27 recites the following:

A computer-readable medium encoded with logic for enhanced extension mobility (*see, e.g.*, Fig. 2; Spec. at 3:4; and 6:2-10:22), the logic when executed operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 3:5-11; 9:19-11:7); and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 11:23-14:2; 14:8-29).

As another example, independent Claim 41 recites the following:

A system for enhanced extension mobility(*see, e.g.*, Fig. 1; Spec. at 3:4; and 6:2-10:22), the system comprising:

means for accessing user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 3:5-11; 9:19-11:7); and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, means for configuring the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 11:23-14:2; 14:8-29).

As another example, independent Claim 42 recites the following:

A system for enhanced extension mobility (*see, e.g.*, Fig. 1; Spec. at 3:4; and 6:2-10:22), the system comprising one or more processing units located at an endpoint and collectively operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 3:5-11; 9:19-11:7);

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users (*see, e.g.*, Fig. 2; Spec. at 11:23-14:2; 14:8-29);

in response to an incoming phone call for the user received at the endpoint, indicate the extension of the user (*see, e.g.*, Spec. at 10:24-11:23);

in response to a request from the user to place an outgoing phone call from the endpoint:

prompt the user to enter the extension of the user prior to placing an outgoing phone call from the endpoint (*see, e.g.*, Spec. at 11:24-12:16); and

generate signaling data for communication with the outgoing phone call that identifies the entered extension of the user (*see, e.g.*, Spec. at 11:24-12:16).

Grounds of Rejection to be Reviewed on Appeal

1. Are Claims 1-8, 10-20, 22-34, and 36-42 properly rejected under 35 U.S.C. § 102(e) as being anticipated by *Sarp*?
2. Are Claims 9, 21, and 35 properly rejected under 35 U.S.C. § 103(a) as being unpatentable over the proposed *Sarp* in view of *Marcus*?
3. Are Claims 43-45 properly re rejected under 35 U.S.C. § 103(a) as being unpatentable over the proposed *Sarp* in view of *Baffes*?

Argument

For at least the following reasons, Appellants respectfully submit that the Examiner's rejections of Claims 1-45 are improper and should be reversed by the Board.

I. The Rejections of Claims 1-8, 10-20, 22-34, and 36-42 under 35 U.S.C. § 102(e) are Improper

A. Overview

Claims 1-8, 10-20, 22-34, and 36-42 stand rejected under 35 U.S.C. § 102(e) as being anticipated by *Sarp*. A copy of *Sarp* is attached as Tab 1 of Appendix B. Appellants respectfully submit that these rejections are improper and should be reversed by the Board.

B. Legal Standard for Demonstrating Anticipation

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987) (emphasis added); M.P.E.P. ch. 2131. "The **identical invention** must be shown in as **complete detail as contained** in the . . . claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989) (emphasis added); *see also* M.P.E.P. ch. 2131. In addition, "[t]he elements must be arranged as required by the claim." *Richardson v. Suzuki Motor Co.*, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989); *In re Bond*, 15 U.S.P.Q.2d 1566 (Fed. Cir. 1990); M.P.E.P. ch. 2131.

The Federal Circuit recently clarified this standard in *Net Moneyin, Inc. v. Verisign, Inc.*, 545 F.3d 1359, 88 U.S.P.Q.2d 1751 (Fed. Cir. 2008). In *Net Moneyin*, the Federal Circuit held that a finding of anticipation under 35 U.S.C. § 102 is proper only when a "reference discloses within the four corners of the document not only all of the limitations claimed but also **all of the limitations arranged or combined in the same way** as recited in the claim." *Net Moneyin*, 545 F.3d at 1371, 88 U.S.P.Q.2d at 1759 (emphasis added). The prior art reference must "**clearly and unequivocally** disclose the claimed invention . . . **without any need for picking, choosing, and combining various disclosures not directly related to each other** by the teachings of the cited reference." *Net Moneyin*, 545 F.3d at 1371, 88 U.S.P.Q.2d at 1760 (emphasis added, internal typographical notations omitted).

C. Argument

1. Independent Claims 1, 15, 27, and 41-42

Claim 1, which Appellants discuss as an example, recites the following:

A system for enhanced extension mobility, the system comprising one or more processing units collectively operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users; and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users.

At least because *Sarp* fails to disclose “each and every element as set forth in [Claim 1],” the rejection of Claim 1 is improper and should be reversed by the Board. *Verdegaal Bros.*, 2 U.S.P.Q.2d at 1053; M.P.E.P. ch. 2131; *see also Net Moneyin.*, 545 F.3d at 88.

As just one example, *Sarp* fails to disclose, teach, or suggest an endpoint capable of supporting a “shared mode according to which the endpoint **concurrently supports** an extension of the user and one or more other extensions of one or more other user,” as recited in Claim 1. In other words, *Sarp* fails to disclose that a single endpoint can support extensions of **multiple users at the same time** (as recited in Claim 1). Rather, *Sarp*, at best, merely discloses a system in which an endpoint may support the extension of **only one** of a number of users at a time (as discussed in further detail below).

The cited portions of *Sarp* disclose a “dynamic telephone configuration” system. *Sarp* at ¶ 7. In a first disclosed embodiment, each of a number of users has a virtual extension associated with physical phone features. *Id.* at ¶ 29. When a user inputs his virtual extension and password into a telephone, the physical port of that telephone is associated with the input virtual extension such that configuration information (i.e., the physical phone features associated with the virtual extension) may be delivered over the physical port and the telephone dynamically configured to support the user. *Id.* at ¶¶ 38-40. In a second disclosed embodiment, each of a number of users has a physical extension rather than a virtual extension. *Id.* at ¶ 49. When a user inputs his physical extension and password into a

telephone, the physical extension of that telephone is swapped with the physical extension input by the user such that configuration information (i.e., the physical phone features associated with the physical extension) associated with the input physical extension may be provided to the dynamically configured telephone. *Id.* at ¶¶ 51-60.

In each of the disclosed embodiments, *Sarp*, at best, only discloses a system in which each of a number of telephones may be configured to support **a single user at a time** (i.e., the user who has entered his virtual/physical extension). *Sarp*, however, does not disclose, teach, or suggest that any of the telephones may support multiple different users at the same time, as would be necessary for *Sarp* to even possibly disclose a “shared mode according to which the endpoint **concurrently supports** an extension of the user and one or more other extensions of one or more other user,” as recited in Claim 1. Therefore, *Sarp* necessarily fails to disclose, teach, or suggest “access[ing] user input indicating a desire of the user to logon at the endpoint in a **shared mode** according to which the endpoint **concurrently supports** an extension of the user and one or more other extensions of one or more other users” and actually “configur[ing] the endpoint to **concurrently support** an extension of the user and one or more other extensions of one or more other users,” as recited in Claim 1.

For at least these reasons, Appellants respectfully submit that the rejections of independent Claim 1 and its dependent claims are improper and should be reversed by the Board. For at least certain analogous reasons, Appellants respectfully submit that the rejection of independent Claims 15, 27, and 41-42 and their dependent claims are improper and should be reversed by the Board.

2. Dependent Claims 4, 16, and 30

Dependent Claims 4, 16, and 30 depend from independent Claims 1, 15, and 27, respectively, which Appellants have shown above to be allowable over *Sarp*. Thus, dependent Claims 4, 16, and 30 are allowable at least because they depend from allowable independent claims. Additionally, dependent Claims 4, 16, and 30 recite further patentable distinctions over *Sarp*.

Dependent Claim 4, which Appellants discuss as an example, recites the following:

The system of Claim 1, wherein the one or more processing units are operable to:

prompt the user to select between a private mode and the shared mode at the endpoint; and

receive a selection by the user of shared mode at the endpoint, the selection providing the accessed user input.

As allegedly disclosing these limitations, the Examiner cites paragraphs 38 and 53-54 of *Sarp*, apparently asserting that *Sarp* discloses prompting a user to select between entering a physical extension (i.e., the claimed “private mode,” according to the Examiner) and a virtual extension (i.e., the claimed “shared mode,” according to the Examiner). *Final Office Action* at 4.

As discussed above with regard to Claim 1, the cited portion of *Sarp* merely discloses alternative embodiments (with virtual extensions being used in one embodiment and physical extensions being used in the other) according to which a telephone may be configured to support a single user. First, because *Sarp* discloses the use of virtual and physical extensions as alternative embodiments, *Sarp* necessarily fails to disclose prompting the user to select between entering a virtual extension and a physical extension. Second, because neither embodiment disclosed in *Sarp* discloses a “shared mode” (for at least the reasons discussed above with regard to Claim 1), *Sarp* is incapable of disclosing that a user is prompted to select a shared mode regardless of what other selection option may be presented to the user. Therefore, *Sarp* fails to disclose, teach, or suggest “prompt[ing] the user to select between a private mode and the shared mode at the endpoint,” as recited in Claim 4.

For at least these reasons, Appellants respectfully submit that the rejection of dependent Claims 4 is improper and should be reversed by the Board. For at least certain analogous reasons, Appellants respectfully submit that the rejection of dependent Claims 16 and 30 are improper and should be reversed by the Board.

II. The Rejection of Claims 9, 21, and 35 under 35 U.S.C. §103(a) is Improper

Claims 9, 21, and 35 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Sarp* in view of *Marcus*. A copy of *Marcus* is attached as Tab 2 of Appendix B.

Appellants respectfully submit that these rejections are improper and should be reversed by the Board.

Claims 9, 21, and 35 depend from independent Claims 1, 15 and 27, respectively, which Appellants have shown above to be allowable over *Sarp*. The Examiner does not allege that *Marcus* makes up for the above-discussed deficiencies of *Sarp*. Thus, dependent Claims 9, 21, and 35 are allowable at least because they depend from allowable independent claims. Additionally, dependent Claims 9, 21, and 35 recite further patentable distinctions over the proposed *Sarp-Marcus* combination. To avoid burdening the record and in view of the clear allowability of independent Claims 1, 15 and 27, Appellants do not specifically discuss these distinctions in this Response. However, Appellants reserve the right to discuss these distinctions in a future Response or on a future Appeal, if appropriate. Moreover, Appellants do not admit that the proposed *Sarp-Marcus* combination is possible or that the Examiner has provided an adequate reason for combining or modifying the references in the manner proposed by the Examiner.

For at least these reasons, Appellants respectfully submit that the rejection of dependent Claims 9, 21, and 35 are improper and should be reversed by the Board.

III. The Rejection of Claims 43-45 under 35 U.S.C. §103(a) is Improper

Claims 43-45 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Sarp* in view of *Baffes*. A copy of *Baffes* is attached as Tab 3 of Appendix B. Appellants respectfully submit that these rejections are improper and should be reversed by the Board.

Claims 43-45 depend from independent Claims 1, 15 and 27, respectively, which Appellants have shown above to be allowable over *Sarp*. The Examiner does not allege that *Baffes* makes up for the above-discussed deficiencies of *Sarp*. Thus, dependent Claims 43-45 are allowable at least because they depend from allowable independent claims. Additionally, dependent Claims 43-45 recite further patentable distinctions over the proposed *Sarp-Baffes* combination. To avoid burdening the record and in view of the clear allowability of independent Claims 1, 15 and 27, Appellants do not specifically discuss these distinctions in this Response. However, Appellants reserve the right to discuss these distinctions in a future Response or on a future Appeal, if appropriate. Moreover, Appellants do not admit that the

proposed *Sarp-Baffes* combination is possible or that the Examiner has provided an adequate reason for combining or modifying the references in the manner proposed by the Examiner.

For at least these reasons, Appellants respectfully submit that the rejection of dependent Claims 43-45 are improper and should be reversed by the Board.

Conclusion

Appellants have demonstrated that, for at least the foregoing reasons, the present invention, as claimed, is clearly patentable over the references cited by the Examiner. Therefore, Appellants respectfully request the Board to reverse the final rejection of the Examiner and instruct the Examiner to issue a Notice of Allowance of all pending claims.

The Commissioner is hereby authorized to charge the amount of \$30.00 for filing this Appeal Brief (the difference between the \$510.00 statutory fee paid in connection with the filing of a previous Appeal Brief in a previous Appeal, for which no final Board decision was reached, and the current statutory fee of \$540.00 - *See* M.P.E.P. § 1204.01) to Deposit Account No. 02-0384 of Baker Botts L.L.P. Although no other fees are believed to be due at this time, the Commissioner is hereby authorized to charge any necessary additional fees and credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Appellants



Chad D. Terrell
Reg. No. 52,279

Date: October 27, 2010

Customer Number: **05073**

Appendix A: The Claims

1. A system for enhanced extension mobility, the system comprising one or more processing units collectively operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users; and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users.

2. The system of Claim 1, wherein one or more of the processing units are located at the endpoint.

3. The system of Claim 1, wherein one or more of the processing units are located at a server remote from the endpoint.

4. The system of Claim 1, wherein the one or more processing units are operable to:

prompt the user to select between a private mode and the shared mode at the endpoint;
and

receive a selection by the user of shared mode at the endpoint, the selection providing the accessed user input.

5. The system of Claim 1, wherein the one or more processing units are operable to:

prompt the user to enter an extension of the user to logon at the endpoint;
access an extension entered by the user; and
configure the endpoint to support the entered extension.

6. The system of Claim 5, wherein the one or more processing units are operable to:

prompt the user to enter a password to logon at the endpoint;
access a password entered by the user;
determine whether the entered password is valid; and
if the entered password is valid, configure the endpoint to support the entered extension.

7. The system of Claim 1, wherein the one or more processing units are further operable, in response to an incoming phone call for the user received at the endpoint, to indicate the extension of the user.

8. The system of Claim 7, wherein the one or more processing units are operable to display the extension of the user at a display screen of the endpoint to indicate the phone call for the user.

9. The system of Claim 7, wherein the one or more processing units are operable to audibly announce a name of the user to indicate the phone call for the user.

10. The system of Claim 7, wherein the one or more processing units are operable to play a ring tone corresponding to the extension of the user to indicate the phone call for the user.

11. The system of Claim 1, wherein the one or more processing units are further operable, in response to a request from the user to place an outgoing phone call from the endpoint, to:

prompt the user to enter the extension of the user prior to placing the outgoing phone call from the endpoint; and

generate signaling data for communication with the outgoing phone call that identifies the entered extension of the user.

12. The system of Claim 1, wherein the one or more processing units are further operable to generate signaling data for communication with every outgoing phone call from the endpoint according to a predetermined extension.

13. The system of Claim 43, wherein the one or more processing units are further operable, in response to the user input indicating a desire of the user to logon at the endpoint in a private mode according to which the endpoint supports only an extension of the user, to configure the endpoint according to one or more preferences of the user.

14. The system of Claim 1, wherein the one or more processing units are further operable, in response to an outgoing phone call from the endpoint, to cause one or more of one or more call detail records (CDRs) and one or more billing records to be updated to indicate a calling extension of the outgoing phone call from the endpoint.

15. A method for enhanced extension mobility, the method comprising:
accessing user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users; and
in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configuring the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users.

16. The method of Claim 15, comprising:
prompting the user to select between a private mode and the shared mode at the endpoint; and
receiving a selection by the user of shared mode at the endpoint, the selection providing the accessed user input.

17. The method of Claim 15, comprising:
prompting the user to enter an extension of the user to logon at the endpoint;
accessing an extension entered by the user; and
configuring the endpoint to support the entered extension.

18. The method of Claim 17, comprising:
prompting the user to enter a password to logon at the endpoint;
accessing a password entered by the user;
determining whether the entered password is valid; and
configuring the endpoint to support the entered extension only if the entered password is valid.

19. The method of Claim 15, further comprising, in response to an incoming phone call for the user received at the endpoint, indicating the extension of the user.

20. The method of Claim 19, comprising displaying the extension of the user at a display screen of the endpoint to indicate the phone call for the user.

21. The method of Claim 19, comprising audibly announcing a name of the user to indicate the phone call for the user.

22. The method of Claim 19, comprising playing a ring tone corresponding to the extension of the user to indicate the phone call for the user.

23. The method of Claim 15, wherein, in response to a request from the user to place an outgoing phone call from the endpoint, the method further comprises:

prompting the user to enter the extension of the user prior to placing an outgoing phone call from the endpoint; and

generating signaling data for communication with the outgoing phone call that identifies the entered extension of the user.

24. The method of Claim 15, comprising generating signaling data for communication with every outgoing phone call from the endpoint according to a predetermined extension.

25. The method of Claim 44, comprising, in response to the user input indicating a desire of the user to logon at the endpoint in a private mode according to which the endpoint supports only an extension of the user, configuring the endpoint according to one or more preferences of the user.

26. The method of Claim 15, further comprising, in response to an outgoing phone call from the endpoint, causing one or more of one or more call detail records (CDRs) and one or more billing records to be updated to indicate a calling extension of the outgoing phone call from the endpoint.

27. A computer-readable medium encoded with logic for enhanced extension mobility, the logic when executed operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users; and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users.

28. The computer-readable medium of Claim 27, being at least partly located at the endpoint.

29. The computer-readable medium of Claim 27, being at least partly located at a server remote from the endpoint.

30. The computer-readable medium of Claim 27, wherein the logic is operable to prompt the user to select between private mode and shared mode at the endpoint, the selection by the user providing the user input.

31. The computer-readable medium of Claim 27, wherein the logic is operable to:
prompt the user to enter an extension of the user to logon at the endpoint;
access an extension entered by the user; and
configure the endpoint to support the entered extension.

32. The computer-readable medium of Claim 31, wherein the logic is operable to:
prompt the user to enter a password to logon at the endpoint;
access a password entered by the user;
determine whether the entered password is valid; and
if the entered password is valid, configure the endpoint to support the entered extension.

33. The computer-readable medium of Claim 27, wherein the logic is further operable, in response to an incoming phone call for the user received at the endpoint, to indicate the extension of the user.

34. The computer-readable medium of Claim 33, wherein the logic is operable to display the extension of the user at a display screen of the endpoint to indicate the phone call for the user.

35. The computer-readable medium of Claim 33, wherein the logic is operable to audibly announce a name of the user to indicate the phone call for the user.

36. The computer-readable medium of Claim 33, wherein the logic is operable to play a ring tone corresponding to the extension of the user to indicate the phone call for the user.

37. The computer-readable medium of Claim 27, wherein the logic is further operable to:

prompt the user to enter the extension of the user prior to placing an outgoing phone call from the endpoint; and

generate signaling data for communication with the outgoing phone call that identifies the entered extension of the user.

38. The computer-readable medium of Claim 27, wherein the logic is further operable to generate signaling data for communication with every outgoing phone call from the endpoint according to a predetermined extension.

39. The computer-readable medium of Claim 45, wherein the logic is further operable, in response to accessing the user input indicating a desire of the user to logon at the endpoint in a private mode according to which the endpoint supports only an extension of the user, to configure the endpoint according to one or more preferences of the user.

40. The computer-readable medium of Claim 27, wherein the logic is further operable, in response to an outgoing phone call from the endpoint, to cause one or more of one or more call detail records (CDRs) and one or more billing records to be updated to indicate a calling extension of the outgoing phone call from the endpoint.

41. A system for enhanced extension mobility, the system comprising:
means for accessing user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users; and

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, means for configuring the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users.

42. A system for enhanced extension mobility, the system comprising one or more processing units located at an endpoint and collectively operable to:

access user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users;

in response to the user input indicating a desire of the user to logon at the endpoint in a shared mode according to which the endpoint concurrently supports an extension of the user and one or more other extensions of one or more other users, configure the endpoint to concurrently support an extension of the user and one or more other extensions of one or more other users;

in response to an incoming phone call for the user received at the endpoint, indicate a the extension of the user;

in response to a request from the user to place an outgoing phone call from the endpoint:

prompt the user to enter the extension of the user prior to placing an outgoing phone call from the endpoint; and

generate signaling data for communication with the outgoing phone call that identifies the entered extension of the user.

43. The system of Claim 1, wherein the one or more processing units are operable to:

access user input indicating a desire of a user to logon at an endpoint in a private mode according to which the endpoint supports only an extension of the user, wherein the user can be concurrently logged on at multiple endpoints; and

in response to the user input indicating a desire of the user to logon at the endpoint in a private mode according to which the endpoint supports only an extension of the user, configure the endpoint to support only an extension of the user.

44. The method of Claim 15, comprising:

accessing user input indicating a desire of a user to logon at an endpoint in a private mode according to which the endpoint supports only an extension of the user, wherein the user can be concurrently logged on at multiple endpoints; and

in response to the user input indicating a desire of the user to logon at the endpoint in a private mode according to which the endpoint supports only an extension of the user, configuring the endpoint to support only an extension of the user.

45. The computer-readable medium of Claim 27, wherein the logic is further operable to:

access user input indicating a desire of a user to logon at an endpoint in a private mode according to which the endpoint supports only an extension of the user, wherein the user can be concurrently logged on at multiple endpoints; and

in response to the user input indicating a desire of the user to logon at the endpoint in a private mode according to which the endpoint supports only an extension of the user, configure the endpoint to support only an extension of the user.

Appendix B: Evidence Appendix

Other than the references listed below and attached as Tabs 1-2 of Appendix B, no evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132, and no other evidence was entered by the Examiner and relied upon by Appellants in the Appeal.

1. *Sarp*
2. *Marcus*
3. *Baffes*

ATTORNEY DOCKET NO.
062891.1251

PATENT APPLICATION
USSN 10/824,180

Appendix B: Evidence Appendix

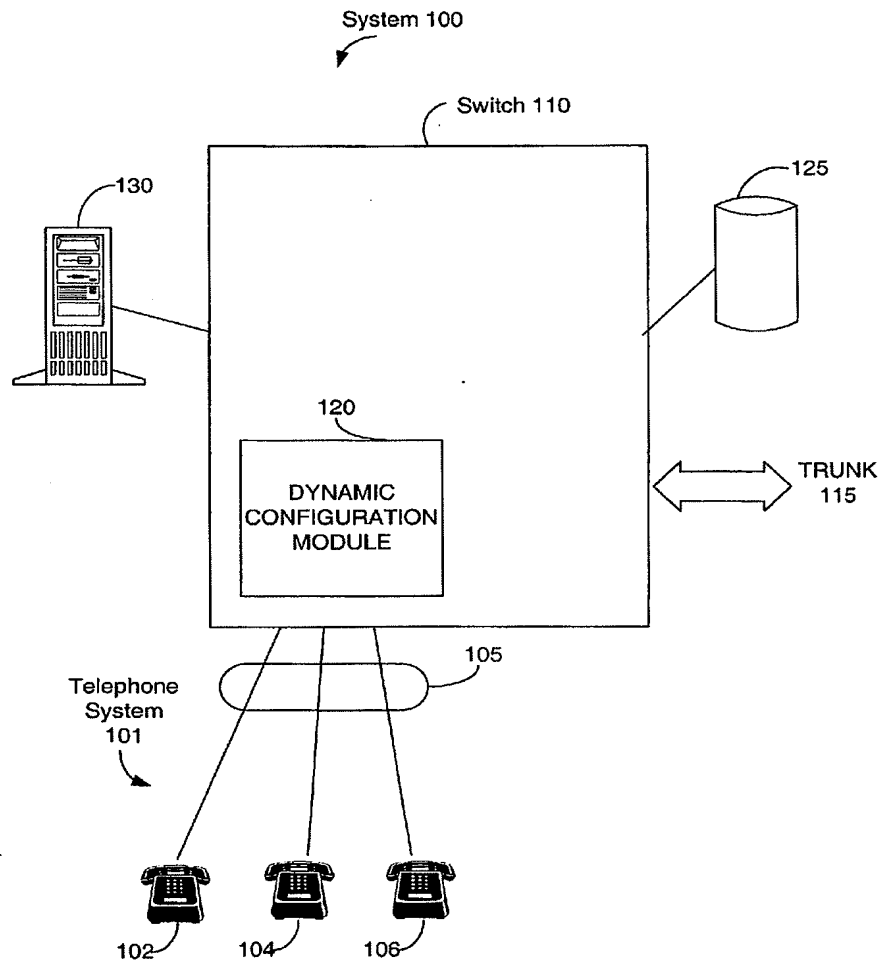
Tab 1 - *Sarp*



US 20050180555A1

(19) **United States**(12) **Patent Application Publication**
Sarp et al.(10) **Pub. No.: US 2005/0180555 A1**(43) **Pub. Date: Aug. 18, 2005**(54) **DYNAMIC TELEPHONE CONFIGURATION****Publication Classification**(75) **Inventors:** S. Murad Sarp, Charlottesville, VA
(US); Angel Gonzalez, Barboursville,
VA (US)(51) **Int. Cl.⁷** H04M 3/42; H04M 7/00(52) **U.S. Cl.** 379/220.01; 379/242**Correspondence Address:**
STERNE, KESSLER, GOLDSTEIN & FOX
PLLC
1100 NEW YORK AVENUE, N.W.
WASHINGTON, DC 20005 (US)(73) **Assignee:** Comdial Corporation, Sarasota, CA
(US)(21) **Appl. No.:** 10/779,671(22) **Filed:** Feb. 18, 2004(57) **ABSTRACT**

The present invention provides a dynamic telephone configuration mode for users of a telephone system. In a virtual configure mode, a user inputs a virtual extension. The input virtual extension is associated with the physical extension of the telephone being dynamically configured. The DC telephone is then operated as if it were associated with the virtual extension input by the user. In the physical configure mode, a user inputs a physical extension. The input physical extension is swapped with the physical extension of the telephone being dynamically configured. The DC telephone is then operated as if it were associated with the physical extension input by the user.



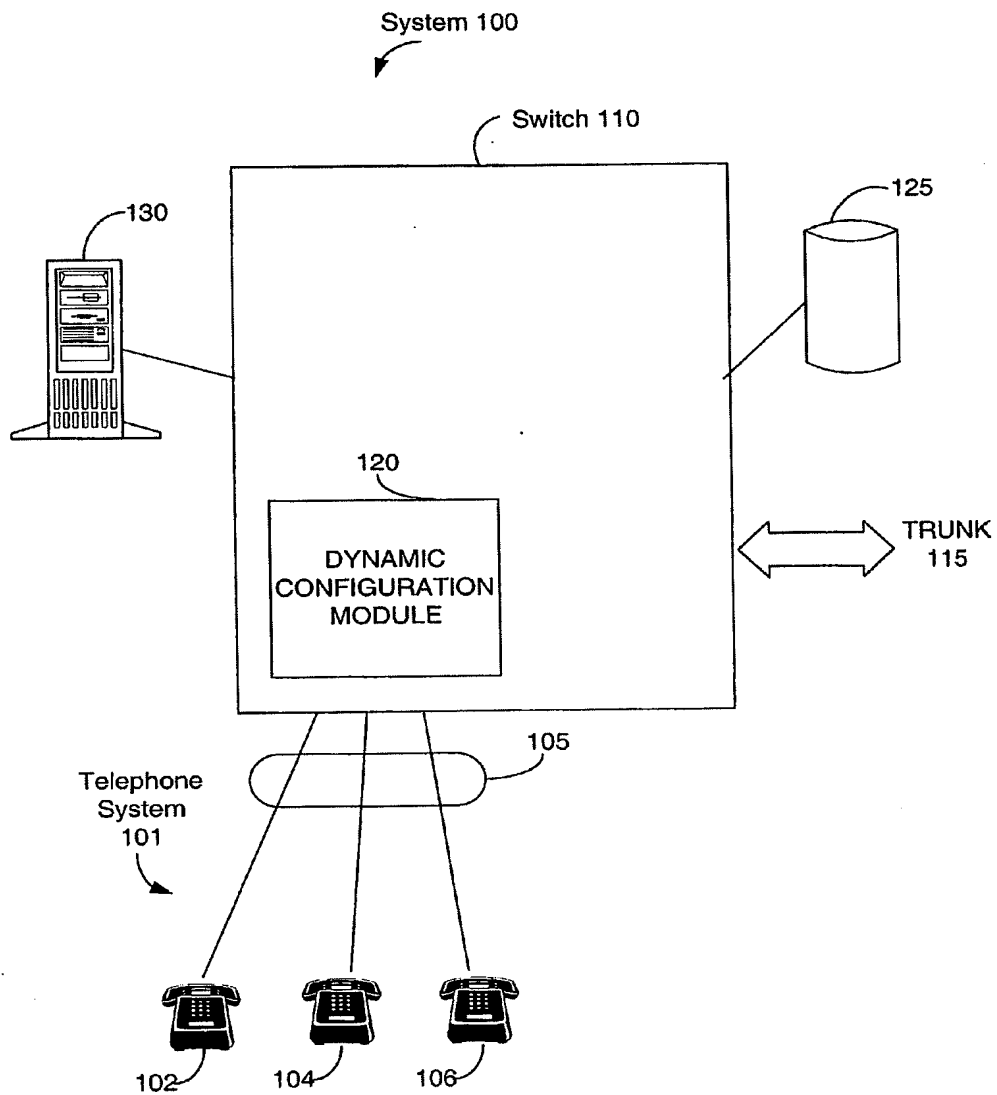


FIG. 1

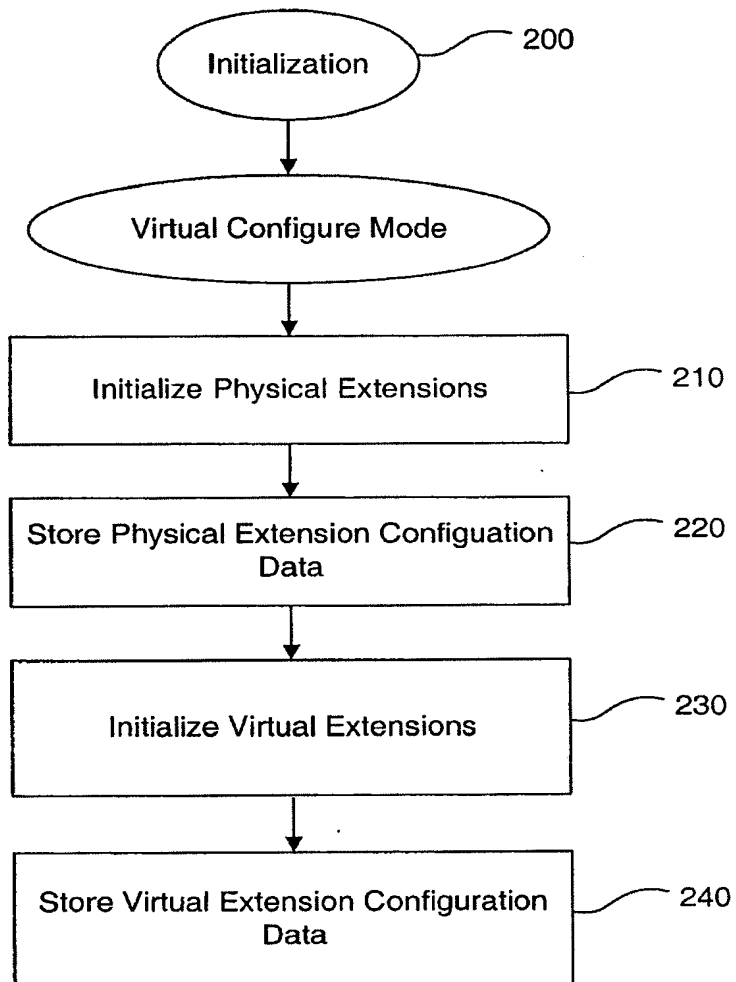


FIG. 2

300

Physical Extensions	Physical Ports	Physical Phone Features including button map with DC button
3002	1	
3004	2	
3006	3	

FIG. 3A

320

Virtual Extensions	User ID	Logical/Physical Mapping	Physical Phone Features including button map with DC button
3002	Jimi	3	
3004	Bob		
3006	Janis		
3008	Ben		

FIG. 3B

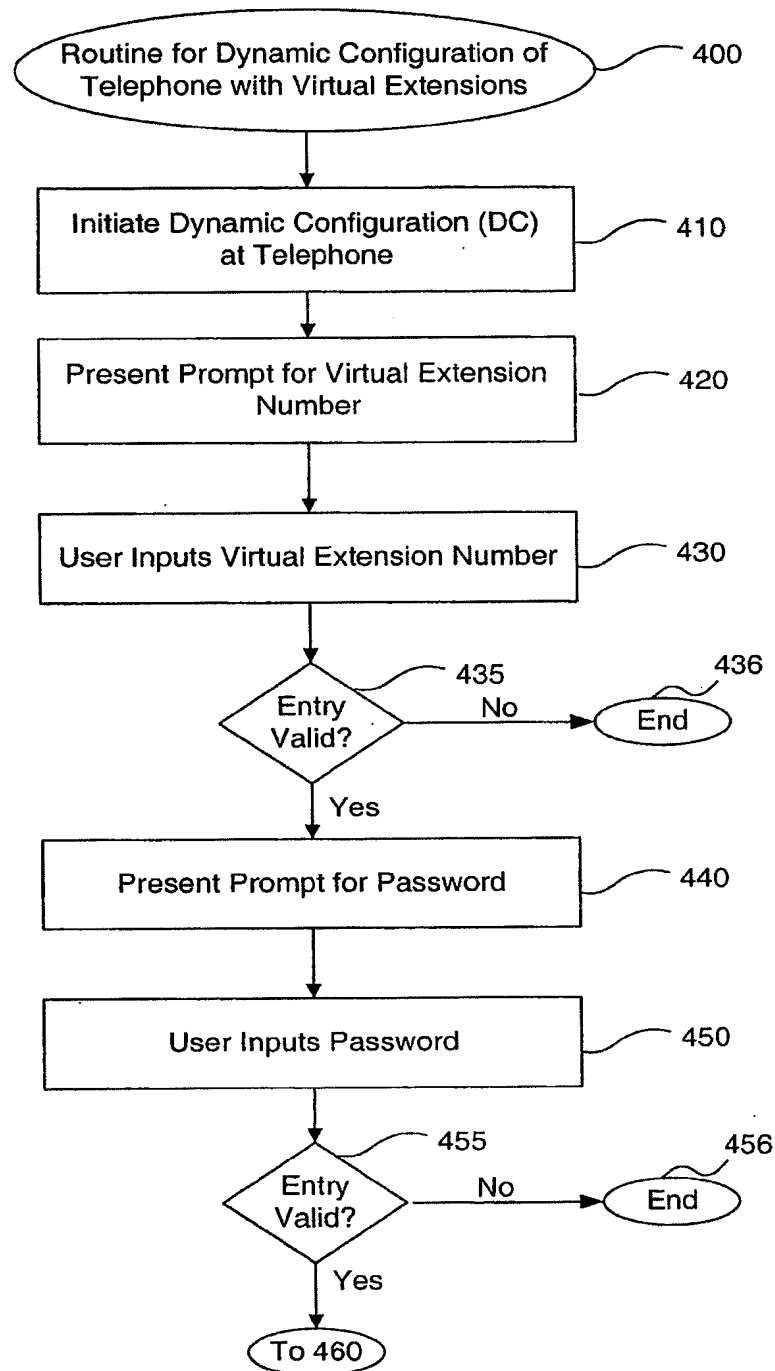


FIG. 4A

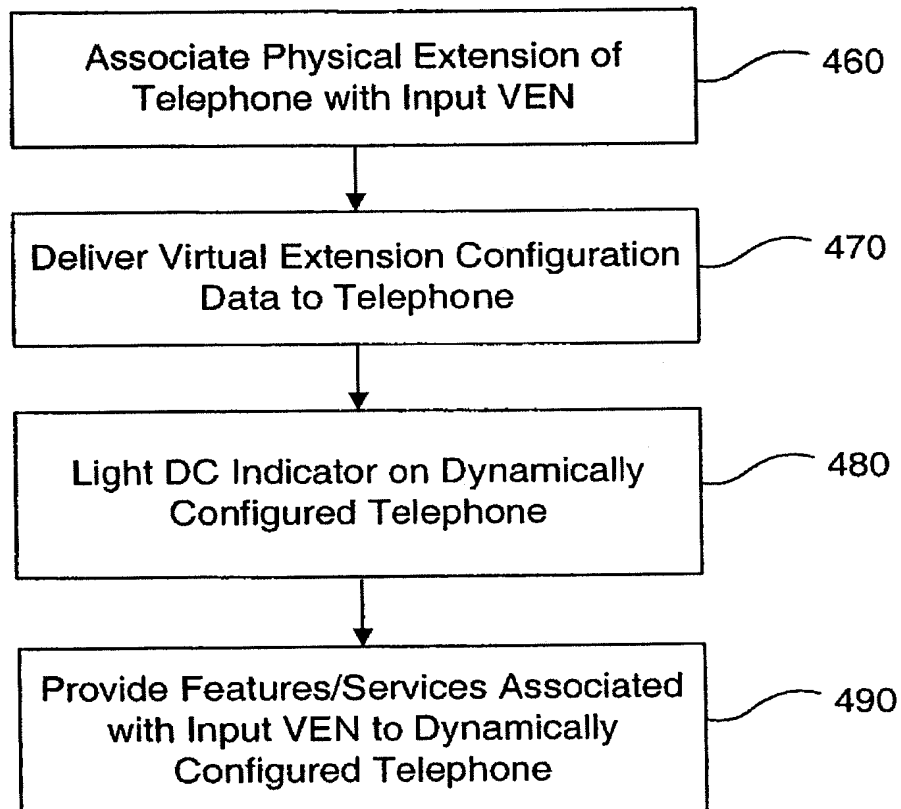


FIG. 4B

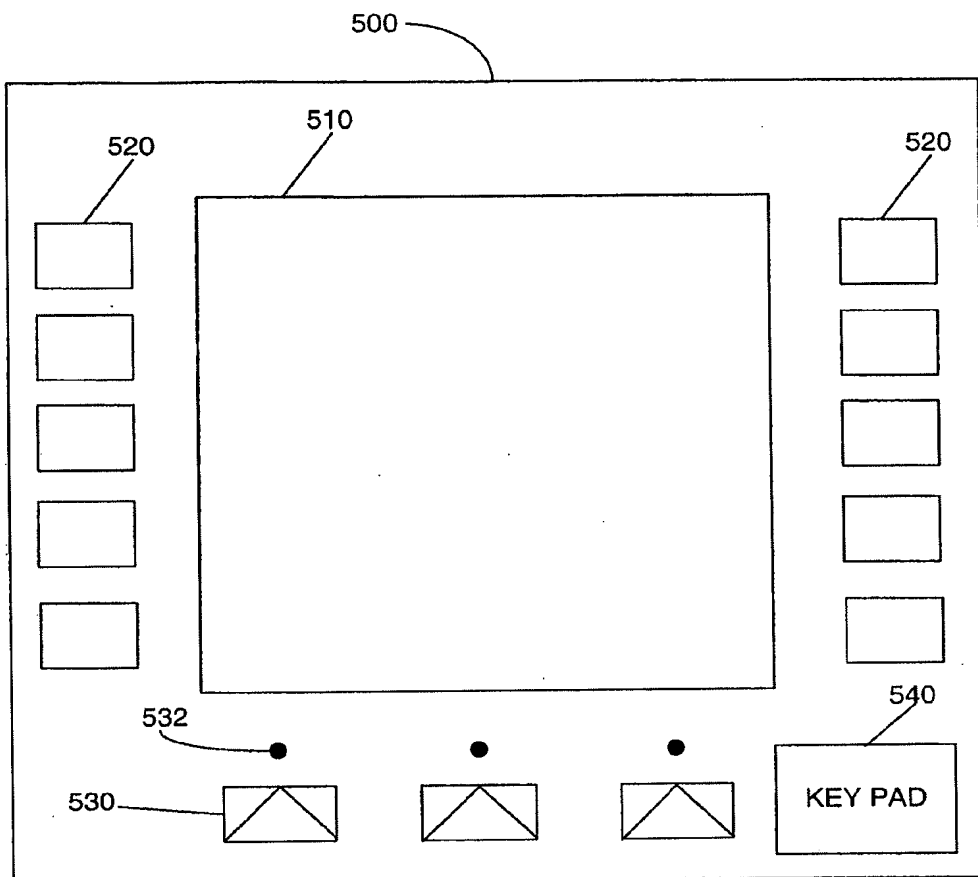


FIG. 5

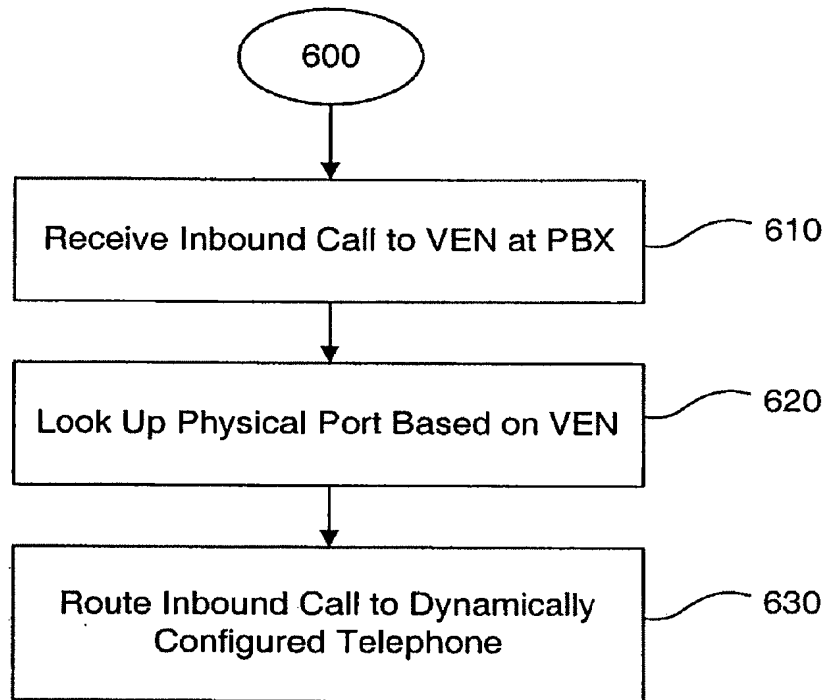


FIG. 6

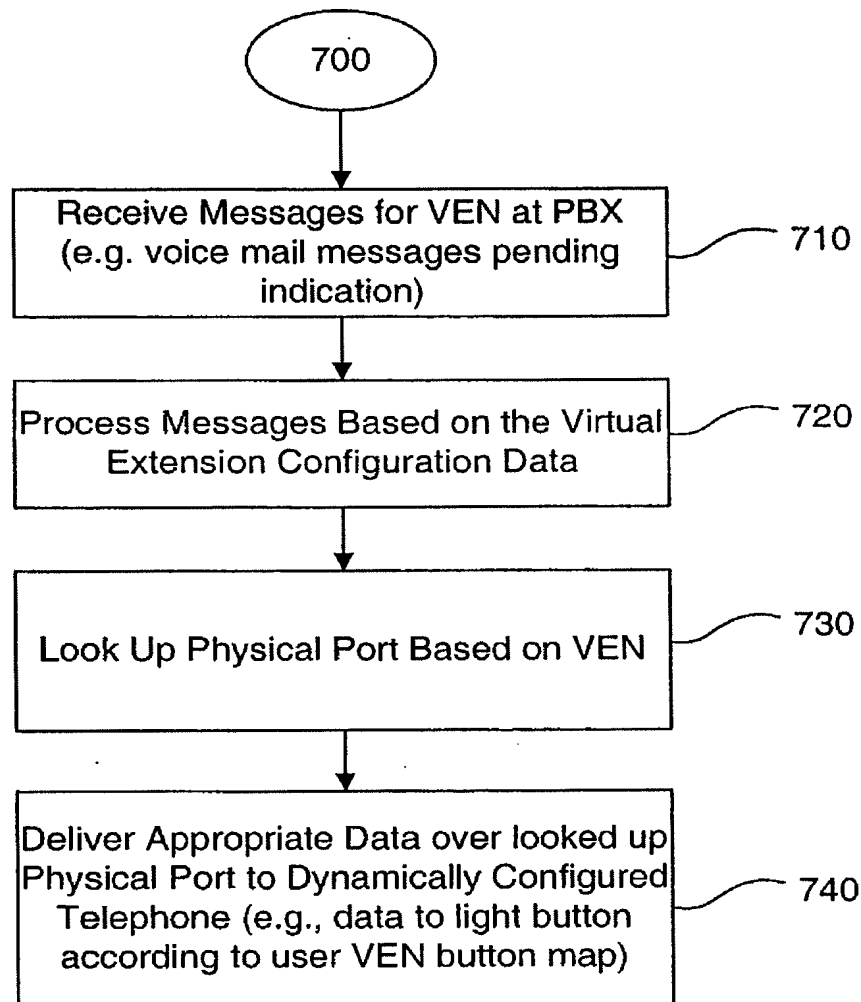


FIG. 7

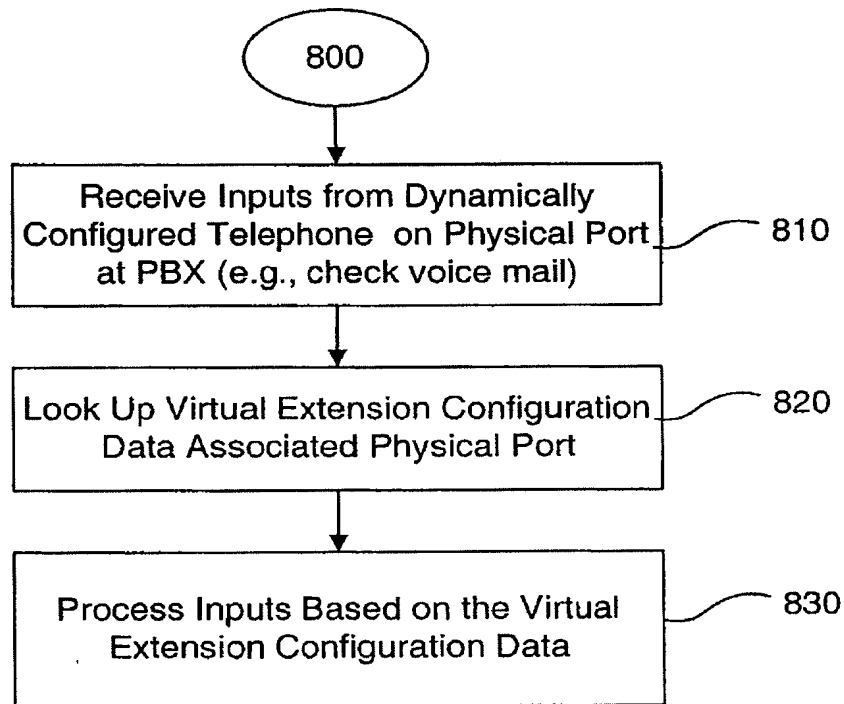


FIG. 8

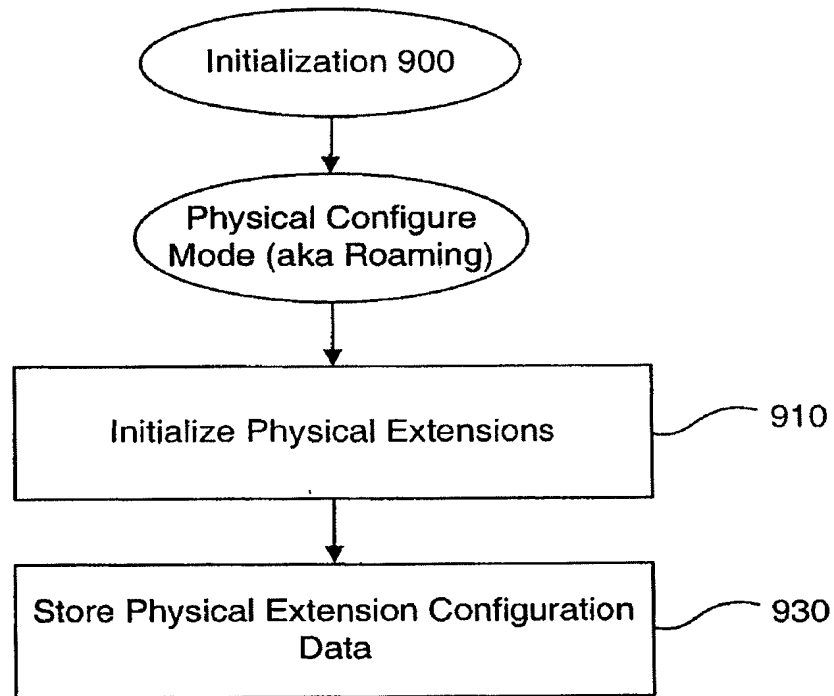



FIG. 9

1010



Physical Extensions	Physical Ports	Physical Phone Features including button map with DC button
3002	3	
3004	2	
3006	1	

FIG. 10

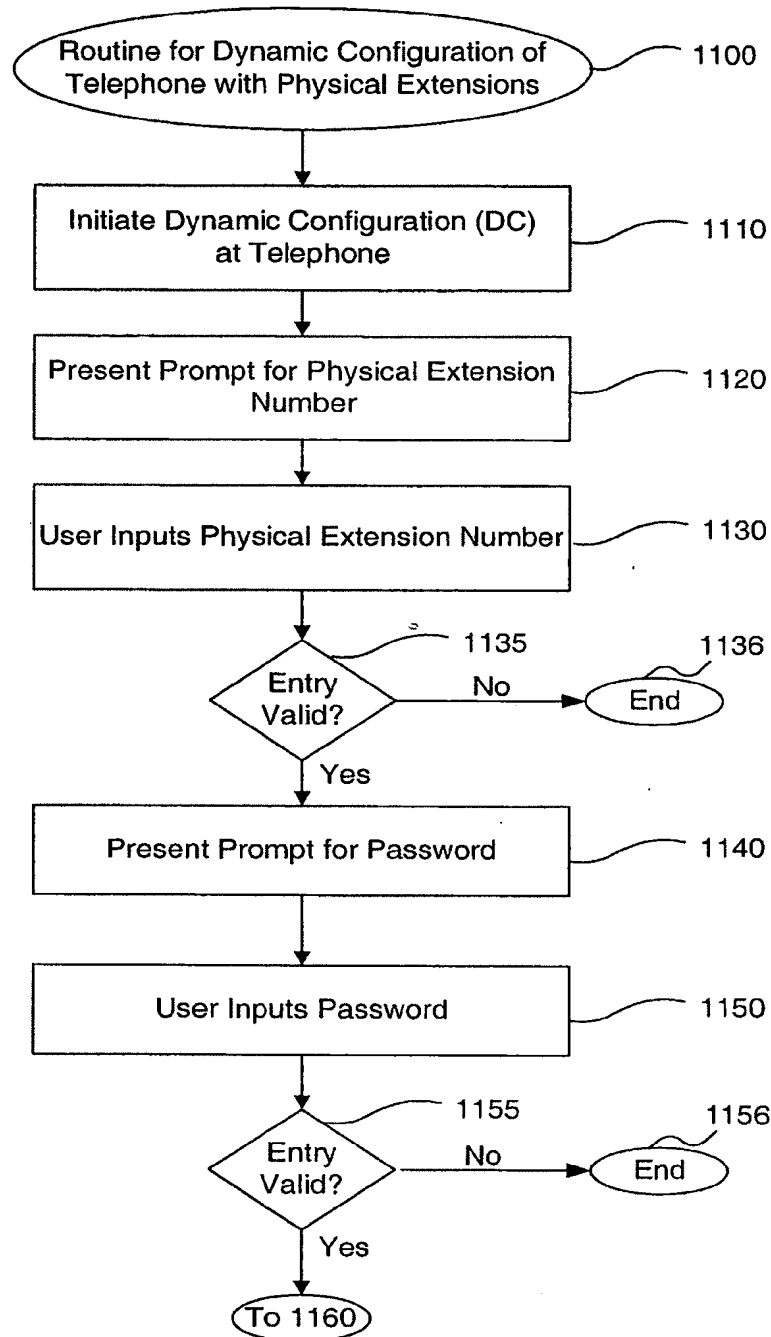


FIG. 11A

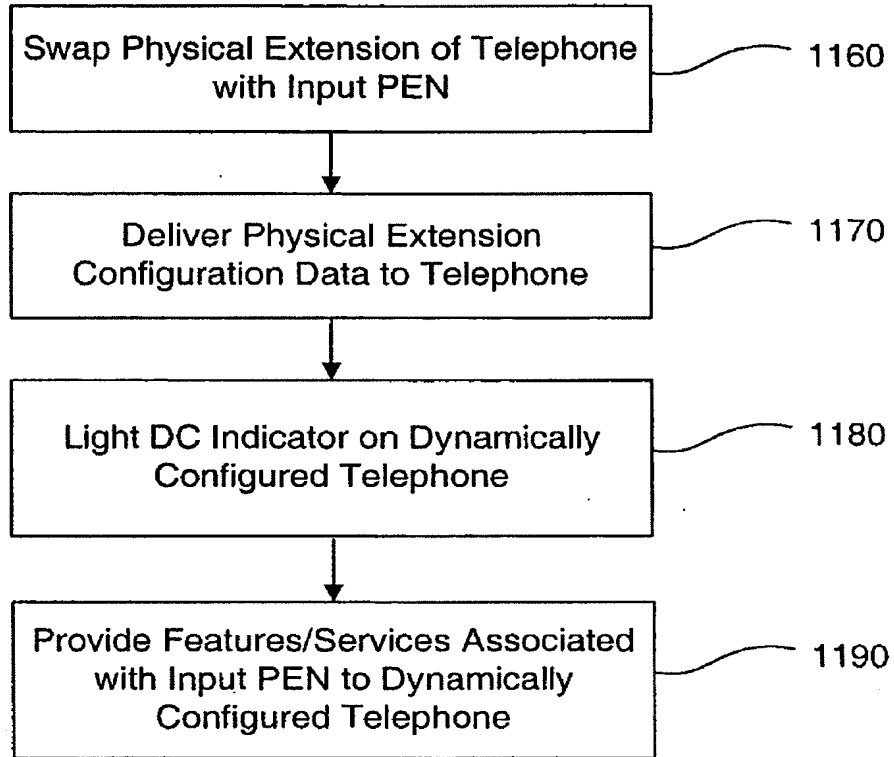


FIG. 11B

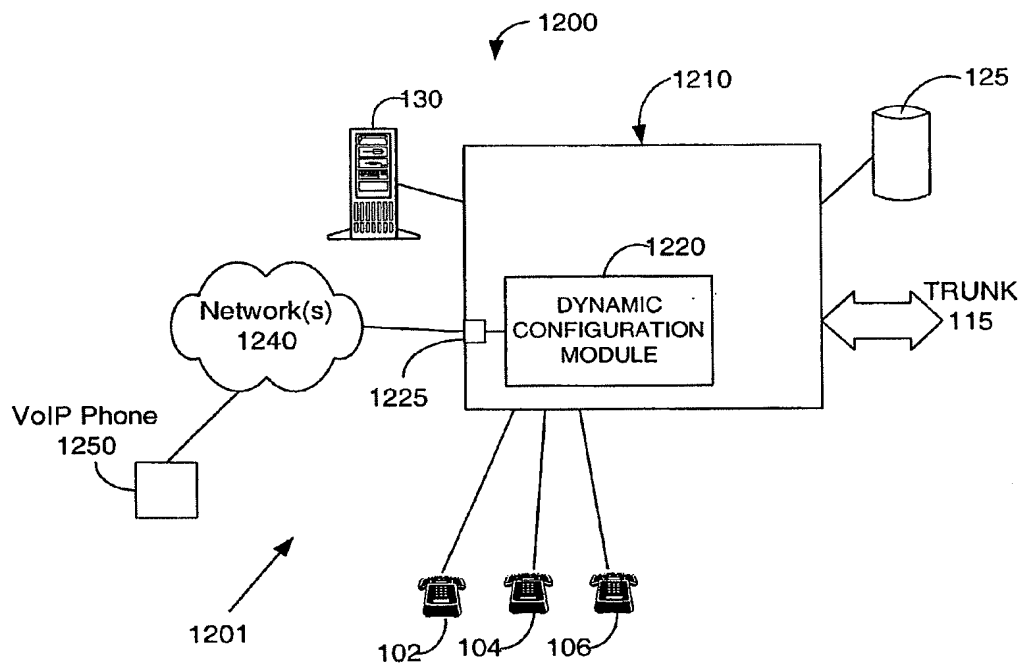


FIG. 12

1300

Virtual Extensions	User ID	Physical Phone Features including button map with DC button
3002	1	
3004	2	
3006	3	
Remote VoIP Phone	Network Port (and IP address of phone)	

FIG. 13

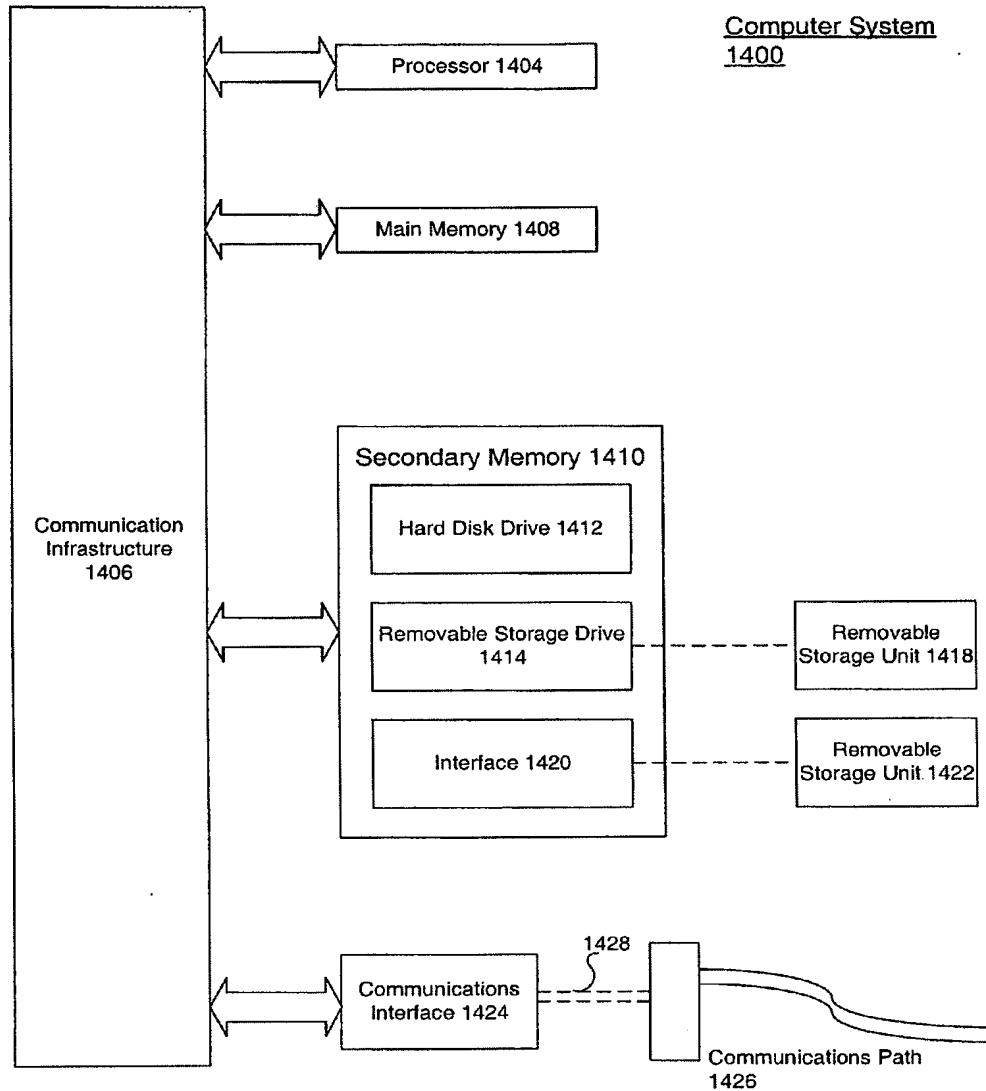


FIG. 14

DYNAMIC TELEPHONE CONFIGURATION

FIELD OF THE INVENTION

[0001] The present invention relates to telecommunications, and in particular to configuring terminal equipment.

BACKGROUND

[0002] A telephone system often has a number of telephones at a subscriber side. These telephones are coupled to a central office switch (CO) or private branch exchange (PBX). One or more trunk lines can couple the CO or PBX to a larger telephone network. A CO or PBX can serve one or more exchanges. An exchange is identified by a dialing prefix (3 digits are used in North America). Each telephone is assigned an extension (4 digits are used for an extension in North America). Typically these extensions are fixed during installation. See, Gilbert Held, *Voice Over Data Networks*, "Telephone Operations," Chapter 5, pp. 113-26 (McGraw-Hill Companies, Inc.: U.S.A. 1998).

[0003] A number of services or features are available in today's telephone systems. These features include call forwarding, ringing, button-map assignments, voice mail and message retrieval delivery, and others. Information identifying the particular features available at a telephone is stored in a memory at the CO or PBX. For example, a table entry associated with a particular extension can include information identifying the features available at the telephone. User settings and choices such as ring style, customized button-map assignments, greeting information and call forwarding instructions are also associated with a particular telephone. Information identifying user settings and features may be stored in memory at the telephone set or at the CO or PBX.

[0004] Once set up, all of this configuration information for a telephone is associated with a particular extension and fixed. While this may be acceptable for a telephone user who uses the same telephone at the same location, it is limiting in a variety of applications and settings. The limitations are especially burdensome when users need to be able to access multiple telephones at multiple locations. The effort a user expends optimizing settings and features for a particular telephone is lost when the user needs to use a different telephone at another location. This can impair productivity as a user may not be able to leverage available telephone features effectively.

[0005] For example, in some workplaces, a user may often work at multiple locations splitting time between a private office and a laboratory. Both of these locations may have telephones; however in many cases, the user may not be able to configure the laboratory telephone because it is shared by multiple users. This can cause frustration and limit productivity as a user may not be able to leverage certain telephone features while in the laboratory at a different extension and telephone.

[0006] Similarly, in many environments, telephones may be used by a number of different people. For example, a realty company may have several different extensions available at an office. Because the number of realtors can exceed the number of telephones, it is often impractical to assign each realtor a dedicated extension and telephone. Instead, the realtors are expected to share telephones on an as needed basis as they are available. Realtors may access the tele-

phones at different times depending upon their schedules, the availability of the telephone or the office associated with the telephone, etc. This means a realtor is unable to customize settings for a particular telephone because he or she must move across different telephones. Configuring a particular extension is then limited to a relatively generic set up that will be acceptable to many different users.

SUMMARY OF THE INVENTION

[0007] The present invention provides dynamic telephone configuration. Embodiments include a virtual configure mode and/or a physical configure mode. In a virtual configure mode, a user inputs a virtual extension. The input virtual extension is associated with the physical extension of the telephone being dynamically configured (DC). The DC telephone is then operated as if it were associated with the virtual extension input by the user. In the physical configure mode, a user inputs a physical extension. The input physical extension is swapped with the physical extension of the telephone being dynamically configured. The DC telephone is then operated as if it were associated with the physical extension input by the user.

[0008] In some embodiments, users can draw upon configuration information at any location within a telephone system. Being able to leverage configuration information from any telephone in a telephone system is especially beneficial to users who need to use a telephone temporarily (such as a worker using a telephone at a temporary workspace) or for several users who share a common pool of telephones (such as workers using desks with telephones on an as needed basis).

[0009] In embodiments, dynamic telephone configuration functionality can be implemented in software, firmware, hardware, or any combination thereof. Further embodiments, features, and advantages of the present invention, as well as the structure and operation of the various embodiments of the present invention, are described in detail below with reference to the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

[0010] The accompanying drawings, which are incorporated herein and form a part of the specification, illustrate the present invention and, together with the description, further serve to explain the principles of the invention and to enable a person skilled in the pertinent art to make and use the invention.

[0011] FIG. 1 is a block diagram of a system for providing dynamic configuration of telephones in a telephone system according to embodiments of the present invention.

[0012] FIG. 2 is a flowchart diagram of a routine for initializing a dynamically configurable switch with physical and virtual extensions according to an embodiment of the present invention.

[0013] FIG. 3A is a table illustrating stored physical extension configuration data.

[0014] FIG. 3B is a table illustrating stored virtual extension configuration data.

[0015] FIGS. 4A-4B are a flowchart diagram of a routine for dynamic configuration of a telephone with virtual extensions according to an embodiment of the present invention.

[0016] FIG. 5 is a diagram of a panel including a display screen for a telephone according to an embodiment of the present invention.

[0017] FIG. 6 is a flowchart diagram that illustrates processing of an inbound call in a dynamically configurable switch according to an embodiment of the present invention.

[0018] FIG. 7 is a flowchart diagram that illustrates processing of inbound messages for services supported by a dynamically configurable switch according to an embodiment of the present invention.

[0019] FIG. 8 is a flowchart diagram that illustrates processing of inputs from a telephone at a dynamically configurable switch according to an embodiment of the present invention.

[0020] FIG. 9 is a flowchart diagram of a routine for initializing a dynamically configurable switch with physical extensions according to an embodiment of the present invention.

[0021] FIG. 10 is a table illustrating swapped physical extension configuration data after roaming.

[0022] FIGS. 11A-11B are a flowchart diagram of a routine for dynamic configuration of a telephone with physical extensions according to an embodiment of the present invention.

[0023] FIG. 12 is a block diagram of a system for providing dynamic configuration of telephones including a remote VoIP telephone in a telephone system according to an embodiment of the present invention.

[0024] FIG. 13 is a table illustrating stored physical extension configuration data including a network port extension according to an embodiment of the present invention.

[0025] FIG. 14 is a block diagram of a computer system platform in an embodiment of the present invention.

[0026] The present invention will now be described with reference to the accompanying drawings. In the drawings, like reference numbers may indicate identical or functionally similar elements. Additionally, the left-most digit(s) of a reference number may identify the drawing in which the reference number first appears.

DETAILED DESCRIPTION OF EMBODIMENTS

[0027] The present invention allows users to dynamically configure telephones in a telephone system.

1. System

[0028] FIG. 1 is a diagram of a system 100 for providing dynamic configuration according to an embodiment of the present invention. System 100 includes a telephone system 101 made up of telephones 102, 104, 106 coupled through links 105 to a switch 110. Switch 110 couples telephone system 101 and trunk 115. Switch 110 can be any type of switch or device for directing inbound and outbound telephone calls to and from telephones 102, 104, 106 and trunk 115. For example, switch 110 can be a private branch exchange (PBX) or a central office (CO) switch.

[0029] Switch 110 includes physical and/or virtual extensions. Each physical extension corresponds to a link to a respective telephone 102, 104, 106. Each virtual extension

corresponds to a respective user of telephone system 101. A provisioning tool 130 is also coupled to switch 110. Provisioning tool 130 can be a control program that enables a system administrator to provision physical and virtual extensions at switch 110. A memory 125 is also coupled to switch 110. Memory 125 stores data that identifies the provisioned physical and virtual extensions and other associated information.

[0030] Switch 110 further includes a dynamic configuration module 120 according to an embodiment of the present invention. Dynamic configuration module 120 can be integral with switch 110 or provided as a separate device coupled to switch 110. Dynamic configuration module 120 associates a physical extension of a telephone with a virtual extension input by user at the telephone. In this way, features associated with the virtual extension input by user can be delivered to the telephone, regardless of an initial provisioning or configuration of the physical extensions by switch 110.

[0031] Dynamic configuration module 120 can be operated in a virtual configure mode or a physical configure mode. The virtual configure mode involves virtual extensions and physical extensions, and is described below with respect to FIGS. 2-8. Physical configure mode involves physical extensions only, and is described below primarily with respect to FIGS. 9-11. In some embodiments, both the virtual configure mode and the physical configure mode can support operation with a remote voice-over IP phone, as described further below with respect to system 1200 in FIG. 12 and FIG. 13. Dynamic configuration module 120 can be implemented in software, firmware, hardware, and/or any combination thereof. An example computer platform supporting a software embodiment is described below with respect to FIG. 14.

2. Virtual Configure Mode

A. Initialization

[0033] FIG. 2 is a flowchart diagram of a routine 200 for initializing a dynamically configurable switch with physical and virtual extensions according to an embodiment of the present invention (steps 210-240). In step 210, physical extensions are initialized. Available physical extensions are associated with available physical ports and associated phone features. For example, in North America, a 4-digit number is used to identify a physical extension. Each physical extension is associated with a particular link to a telephone. In the simple example of three telephones 102, 104, 106 in FIG. 1, three different 4-digit extensions (3002, 3004, 3006) can be assigned. Physical ports in switch 110 that lead to corresponding telephones 102, 104, 106 are then associated with the respective physical extensions. Physical phone features may also be associated with the physical extensions. Such physical phone features can be any phone system setting, user setting or service choice, including but not limited to, button-map assignments, ring style, call forwarding, call waiting, transfer, hold, greeting style, etc.

[0034] In step 220, the initialized physical extension configuration data is stored in memory. For example, memory 125 in FIG. 1 can be a database that stores physical extension configuration data. FIG. 3A shows, in tabular form, physical extension configuration data, including but not limited to physical extensions, physical ports, and physi-

cal phone feature information. In the example shown in FIG. 3A, Table 300 includes a first entry having fields that associates physical extension 3002 with physical port 1 of switch 110 and physical phone feature information. A second entry is provided that associates physical extension 3004 with physical port 2 and physical phone feature information. A third entry is provided for physical extension 3006 and associated physical port 3 and physical phone feature information. Referring to FIG. 1, in one implementation, memory 125 is a database, and provisioning tool 130 is an application that supports Virtual Mail Manager Interface (VMMI) programming. A system administrator using provisioning tool 130 can be provided with a series of control menus through a graphical user interface. Such control menus enable the system administrator to easily identify and/or edit physical extensions (e.g., add/modify/delete), associated ports and associated physical phone features in carrying out initialization step 210. Records can be generated for each provisioned physical extension and can be stored in the database in memory 125.

[0035] Returning to FIG. 2, in step 230, virtual extensions are initialized. Virtual extensions are initialized similarly to the physical extensions described above with respect to step 210. For example, in step 230, virtual extensions are associated with physical phone features. Virtual extensions, however, are generally associated with a user of telephone system 101. Virtual extensions can also be identified by a 4-digit number and further associated with a user identification (ID), such as a user name. The physical phone features associated with virtual extensions can be any phone system setting, user setting or service choice, including but not limited to, button-map assignments, ring style, call forwarding, call waiting, transfer, hold, greeting style, etc. According to an embodiment of the present invention, mapping information can also be associated with virtual extensions. Such mapping information is used to map a virtual extension to a physical port at a telephone during the dynamic configuration of the telephone. This mapping is described further below with respect to routine 400 in FIG. 4.

[0036] In step 240, virtual extension configuration data is stored. In the embodiment of FIG. 1, virtual extension configuration data is stored in memory 125. FIG. 3B shows, in tabular form, virtual extension configuration data, including but not limited to virtual extensions, user ID, mapping information, and physical phone feature information. In the example in FIG. 3B, Table 320 includes a first entry having fields that associates virtual extension 3002 with user Jimi of switch 110 and physical phone feature information. A second entry is provided that associates virtual extension 3004 with user Bob and physical phone feature information. A third entry is provided for virtual extension 3006 with user Janis and physical phone feature information. A fourth entry is provided for virtual extension 3008 with user Ben and physical phone feature information. In the embodiment of FIG. 1, each virtual extension includes mapping information for logically mapping the virtual extension to a physical port as described further below with respect to FIG. 4.

[0037] B. Operation

[0038] FIGS. 4A and 4B are a flowchart diagram of a routine 400 for dynamic configuration of a telephone with virtual extensions according to an embodiment of the

present invention (steps 410-490). In step 410, dynamic configuration (DC) at a telephone is initiated. For example, although a user may have access to a telephone, such as telephone 106 in telephone system 101 in FIG. 1, the telephone 106 may not be configured to provide the user access to one or more features available to the telephone 106. The present invention allows the user to initiate dynamic configuration at telephone 106. For example, the user can press a particular DC button or a combination of buttons at telephone 106. In some embodiments, initiation of dynamic configuration may be voice-automated. For example, the user may provide a voice command to initiate dynamic configuration of the telephone 106. Once dynamic configuration is initiated, dynamic configuration module 120 presents a prompt for a virtual extension number (step 420). The user then inputs the virtual extension number to satisfy the prompt (step 430). For example, the prompt may appear in a display at telephone 106. The user then enters through the keypad at telephone 106 his or her virtual extension.

[0039] Returning to FIG. 4, a check is made at step 435 to determine whether the virtual extension entry is valid. In an embodiment, this check is carried out by dynamic configuration module 120 in FIG. 1 in response to the information input by the user at telephone 106. If the virtual extension number is not a valid extension number, then control proceeds to step 436 and routine 400 ends. If the entry is valid, then control proceeds to step 440. In step 440, dynamic configuration module 120 presents a prompt for a password to the user at telephone 106. For example, dynamic configuration module 120 can send instructions to telephone 106 to display a prompt to the user to input a password. In step 450, user responds to the prompt and inputs a password for telephone 106. As with step 430, the user can use the keypad of the telephone 106 to input a password. Dynamic telephone 106 then sends this input to dynamic configuration module 120. Dynamic configuration module 120 checks the password entry to determine whether it is valid. If the entry is not valid, then control proceeds to step 456 and routine 400 ends. If the entry is valid, then control proceeds to step 460 (see FIG. 4B).

[0040] In step 460, dynamic configuration module 120 associates the physical extension of a telephone from which the valid virtual extension number and password are received, in steps 430 and 450, with the virtual extension number input in step 430. For instance, consider the example where a user, Jimi, with virtual extension 3002 initiates dynamic configuration at telephone 106. Telephone 106 is coupled to switch 110 at a physical port 3 as shown in table 300. Dynamic configuration module 120 then looks up physical port 3 to determine physical extension 3006. Since Jimi inputs his virtual extension number 3002, dynamic configuration module 120 associates virtual extension 3002 with physical port 3, as shown in Table 320 in FIG. 3B. Dynamic configuration module 120 in FIG. 1 may write mapping information into a field in an appropriate record in database 125, so that the virtual extension 3002 is now associated with physical port 3.

[0041] In step 470, virtual extension configuration data is then delivered over the physical port to the dynamically configured telephone. For example, prior to Jimi dynamically configuring telephone 106, the configuration data at the telephone was probably either a default configuration or a configuration of another user. Because of Jimi's initiating

dynamic configuration, virtual extension configuration data associated with virtual extension 3002 is then sent to telephone 106 through port 3. Such virtual extension configuration data can include data needed for telephone 106 to support the particular phone features associated with the virtual extension. FIG. 4 is illustrative and not intended to limit the present invention. For example, the virtual configuration data can be stored in a memory coupled to the switch in some embodiments.

[0042] In step 480, dynamic configuration module 120 in FIG. 1 sends a signal to indicate to telephone 106 that a dynamic configuration indicator should be lit on telephone 106. For example, an indicator above the dynamic configuration button can be lit to notify the user that the dynamic configuration is complete. This example is illustrative only. Other types of indicators can be used.

[0043] FIG. 5 is a diagram of a panel 500 that includes a display screen 510 for a telephone, such as telephone 106 in FIG. 1, according to an embodiment of the present invention. Panel 500 also includes buttons 520 for activating different phone features, such as call waiting, multiple lines, forwarding, hold, transfer, or call conferencing. Panel 500 includes a keypad 540 that may include numeric characters and any other characters for carrying out telephone operation. Additional buttons may also be provided for special features. As shown in FIG. 5, a dynamic configuration button 530 is provided to enable a user to easily initiate dynamic configuration in step 410, as described above. An indicator 532 such as a visible light-emitting diode is provided near button 530. For instance, indicator 532 can be lit as described with respect to step 480 when dynamic configuration has been carried out. Panel 500 is illustrative and not intended to limit the present invention.

[0044] Referring again to FIG. 4, in step 490, features associated with the virtual extension input in step 430 are provided to the dynamically configured telephone. User, Jimi, at telephone 106 in FIG. 1, then receives features associated with the virtual extension to which he may be accustomed. At this point, switch 110 operates in a conventional manner to support and deliver telephone features and services to telephone 106, as if the user Jimi and his virtual extension number were initially configured as being associated a physical extension at telephone 106. Step 490 is described further below with respect to specific operations in FIGS. 6-8.

[0045] FIG. 6 is a flowchart diagram 600 that illustrates processing of an inbound call in a dynamically configurable switch 110 in FIG. 1 according to an embodiment of the present invention. In step 610, an inbound call to a virtual extension number (VEN) is received over trunk 115 at switch 110. Switch 110 then looks up the physical port based on the virtual extension number identified in the inbound call (step 620). This lookup includes looking at the mapping information provided by dynamic configuration module 120, as described above with respect to step 460. Switch 110 then routes the inbound call over a physical port to the dynamically configured telephone currently being used by a user associated with the virtual extension. In this way, once the user has dynamically configured a telephone based on his or her virtual extension, switch 110 will provide inbound telephone calls to the user at the dynamically configured telephone. Switch 110 can also provide other features or

messages such as voicemail message indications, as well, as will be described further below with respect to FIG. 7.

[0046] FIG. 7 is a flowchart diagram that illustrates a routine 700 for processing inbound messages for services supported by a dynamically configurable switch 110 in FIG. 1, according to an embodiment of the present invention. In step 710, switch 110 receives messages associated with a virtual extension. For example, such messages may be sent by a voicemail application to notify a user that a voicemail message has been received and is waiting for retrieval. Switch 110 then processes the received messages based on the virtual extension configuration data (step 720). Switch 110 looks up the physical port based on associated virtual extension data. As described with respect to step 620 above, this physical port that is looked up has been associated during dynamic configuration with a telephone at which the user having the virtual extension is now present. In step 740, switch 110 then delivers appropriate data over the looked up physical port to the dynamically configured telephone. For example, switch 110 may send a command to light a voicemail light in accordance with particular button map assignments associated with a user of the virtual extension.

[0047] FIG. 8 is a flowchart diagram that illustrates a routine 800 for processing inputs made by a user at the telephone, once the telephone has been dynamically configured (steps 810-830). In step 810, switch 110 in FIG. 1 receives inputs from a dynamically configured telephone on a particular physical port of switch 110. For example, the user may press a button which corresponds to a command to retrieve voicemail. Switch 110 then looks up virtual extension configuration data (e.g., physical phone feature information in the field shown in FIG. 3B) associated with the physical port on which the inputs are received (step 820). Based upon this lookup, switch 110 processes the received inputs based on the virtual extension configuration data (step 830). For example, switch 110 may look up the virtual extension configuration data and determine that the particular button command sent by the user is associated with voice message retrieval. Switch 110 will then forward the user's command to retrieve voicemail for the appropriate voicemail application. Such a voicemail application, for example, can be running locally at switch 110, or remotely in another part of the telephone network in telephone system 101.

3. Physical Configure Mode

[0048] A. Initialization

[0049] FIG. 9 is a flowchart diagram of a routine 900 for initializing a dynamically configurable switch with physical extensions, according to an embodiment of the present invention. In step 910, physical extensions at switch 110 are initialized. Similar to the virtual configure mode initialization described above with respect to step 210, this initialization can include initializing physical extensions, physical ports, and physical phone features for switch 110. Physical extension configuration data is then stored (step 920). For example, step 910 can be performed by a system administrator using provisioning tool 130. Control menus and other screen displays can be provided to enable user to add or modify data relating to provisioning of physical extensions. In one embodiment, a user interface is provided that assists a system administrator with VMMI programming of a database stored at memory 125. This database can be used

to store physical extension configuration data in step 920. In one example, a table 300 can be generated and stored, as described above with respect to FIG. 3A.

[0050] B. Operation

[0051] In the physical configure mode, a user inputs a physical extension. The input physical extension is swapped with the physical extension of the telephone being dynamically configured (i.e. the DC telephone). The DC telephone is then operated as if it were associated with the physical extension input by the user.

[0052] For example, referring to FIG. 1, a user may be present at telephone 106 in telephone system 101, rather than at telephone 102 in telephone system 101. The user (or system administrator) may have already configured telephone 102 to make optimal use of certain features available to telephone 102. Telephone 102, for example, may be at a desk in a cubicle in which the user normally works. While near telephone 106, however, the user may desire to take advantage of the features configured previously with respect to telephone 102. The present invention allows the user to initiate dynamic configuration at telephone 106. For example, the user can press a particular DC button at telephone 106 to initiate dynamic configuration.

[0053] FIGS. 11A and 11B are a flowchart diagram of a routine 1100 for dynamic configuration of a telephone with physical extensions (steps 1110-1190). Routine 1100 operates similarly to routine 400 described above, except that virtual extensions are not used. In step 1110, a user initiates a dynamic configuration at a telephone. For example, a user at telephone 106 can press a dynamic configuration button. In step 1120, a prompt is presented for a user's physical extension. For example, dynamic configuration module 120 can send a command to instruct telephone 106 to display a prompt for the user's physical extension.

[0054] In step 1130, the user inputs his or her physical extension. The user can input the physical extension through a keypad on a panel of telephone 106.

[0055] Dynamic configuration module 120 then receives the input physical extension and performs a check to determine if the entry is valid (step 1135). If the physical extension entry is invalid, in that it is not a physical extension associated with telephone system 101 or it is in an improper format or other error, then control proceeds to step 1136, and routine 1100 is ended. If the entry is valid, then control proceeds to step 1140. In step 1140, a user is presented with a prompt for a password. For example, dynamic configuration module 120 in FIG. 1 can send a command to telephone 106 to display a prompt for password on a panel. In step 1150, a user inputs a password. For example, a user at telephone 106 can input a password using a keypad on a panel of telephone 106.

[0056] In step 1155, a check is made to determine if the password entry is valid. For example, dynamic configuration module 120 can evaluate the password inputted at telephone 106 and sent to dynamic configuration module 120. If the password is not valid, then control proceeds to step 1156, and routine 1100 ends. If the entry is valid, then control proceeds to step 1160.

[0057] As shown in FIG. 11B, in step 1160 the physical extension of the telephone to be dynamically configured is

swapped with the physical extension input by the user in step 1130. Dynamic configuration module 120 swaps the physical port information associated with the physical extension of the telephone to be configured and the physical extension input by the user. FIG. 10 shows in tabular form how physical port information may be changed in step 1160. Table 1010 is identical to table 300 described previously, except that the physical ports for physical extensions 3002 and 3006 have been swapped. For example, this would occur when a user (Jimi) inputs physical extension 3002 in step 1130 at telephone 106 associated with physical extension 3006. In step 1160, then, dynamic configuration module 120 swaps the physical port 3 information formerly associated with physical extension 3006, for the physical port 1 information associated with the physical extension 3002 input by the user.

[0058] Steps 1170-1190 then proceed similar to steps 470-490 in FIG. 4B, described above. In step 1170, physical extension configuration data is delivered to the telephone that is being dynamically configured. Such physical extension configuration data can include information relating to the physical phone features, including but not limited to button map assignments, call forwarding, call waiting, forward, transfer and hold, and other features and/or services of telephone network 101. FIG. 11 is illustrative and not intended to limit the present invention. For example, the physical configuration data can be stored in a memory coupled to the switch in some embodiments.

[0059] In step 1180, a dynamic configuration indicator is lit on the dynamically configured telephone. For example, dynamic configuration module 120 can send a command to telephone 106 to indicate that the telephone 106 has been dynamically configured. In step 1190, features associated with the input physical extension are then provided to the dynamically configured telephone. In other words, switch 110 operates to process inbound calls and inbound messages and inputs from the telephone, as described previously with respect to step 490 and FIGS. 6-8.

[0060] For example, to process inbound calls directed to the physical extension input by the user, switch 110 looks up physical port information associated with the physical extension and then routes the calls accordingly. In this way, inbound calls are routed to a dynamically configured telephone, based on the physical extension input by the user during dynamic configuration. Similarly, messages received from applications such as voicemail applications or other telephone service applications that are directed to a physical extension, are similarly processed by switch 110. Switch 110 identifies physical port information associated with the physical extension and delivers it to the dynamically configured telephone accordingly.

[0061] Inputs made at the dynamically configured telephone are likewise received at a physical port associated with the dynamically configured telephone. Switch 110, however, looks up the physical extension associated with the physical port, and because of the swapping described with respect to step 1160, processes the inputs based on the appropriate phone features and other configuration data associated with the physical extension input by the user during dynamic configuration.

4. Remote VoIP Phone

[0062] According to a further feature of the present invention, dynamic configuration can also be carried out for telephones with a telephone system that includes remote phones. Referring to FIG. 12, system 1200 includes a switch 1210 and a dynamic configuration module 1220. Switch 1210 and dynamic configuration module 1220 operate similarly to switch 110 and dynamic configuration module 120 respectively, described in detail above. For example, system 1200 includes provisioning tool 130 and memory 125 coupled to switch 1210. Switch 1210 couples telephones in a telephone system 1201 to trunk 115. In this embodiment, telephone system 1201 also includes a voice over IP (VoIP) phone 1250 coupled to switch 1210 over a network 1240 through port 1225 in switch 1210. Switch 1210 then includes physical ports for coupling telephones 102, 104, 106 and port 1225 for coupling a remote VoIP phone 1250. In one example, port 1225 is any type of communication port including but not limited to a network port for coupling to a network, such as the Internet.

[0063] System 1200 operates like system 100 in FIG. 1 described above. One difference is provisioning tool 130 provisions physical extensions for each of the physical ports coupled to telephones 102, 104 and 106, and further provisions a physical extension for the remote phone 1250. In this case, the provisioning is made for a network port. Otherwise, switch 1210 and dynamic configuration module 1220 can operate to support a virtual configure mode and/or a physical configure mode, as described above with respect to FIGS. 2-11.

[0064] FIG. 13 shows, in tabular form, virtual extension configuration data, including but not limited to virtual extensions, user IDs, and physical phone feature information. In the example shown in FIG. 13, Table 1300 includes a first entry having fields that associates virtual extension 3002 with user ID 1 and physical phone feature information. A second entry is provided that associates virtual extension 3004 with user ID 2 and physical phone feature information. A third entry is provided for virtual extension 3006 and associated user ID 3 and physical phone feature information. A fourth entry is provided that associates a remote VoIP phone with a network port and physical phone feature information. The remote VoIP phone may be associated with an IP address of a telephone 102, 104, 106 in the telephone system 101, for example.

[0065] The present invention can be implemented in any telephone system, including but not limited to a telephone system FXII coupled to IPRIMO telephones both available from Comdial Corporation, a Delaware corporation. Telephones 102, 104, 106 are illustrative and not intended to limit the present invention. Telephones 102, 104, 106 can be any type of telephone, including but not limited to, a digital telephone available from Comdial such as an Impact telephone, an internet protocol (IP) telephone such as an IPRIMO telephone, or other type of telephone or terminal equipment. As used herein, the term "telephone" refers to any type of terminal equipment or device for coupling at least voice and/or data to a telephone system.

[0066] Switches 110, 1210 can be any type of switch including but not limited to a private branch exchange (PBX) or central office switch. Trunk 115 is illustrative, and

in general the present invention can be used with any type and any number of trunk lines depending upon the capacity of switches 110, 1210.

5. Computer System Platform

[0067] The present invention including dynamic configuration modules 120, 1220 may be implemented using hardware, software or a combination thereof and may be implemented in a computer system or other processing system. In an embodiment, the invention is directed toward a computer program product executing on a computer system capable of carrying out the functionality described herein, particularly that of dynamic configuration modules 120, 1220. An example of a computer system 1400 is shown in FIG. 14. The computer system 1400 includes one or more processors, such as processor 1404. The processor 1404 is connected to a communication infrastructure 1406, such as a communication bus. Various software embodiments are described in terms of this example computer system. After reading this description, it will become apparent to a person skilled in the relevant art how to implement the invention using other computer systems and/or computer architectures.

[0068] Computer system 1400 also includes a main memory 1408, preferably random access memory (RAM), and may also include a secondary memory 1410. The secondary memory 1410 may include, for example, a hard disk drive 1412 and/or a removable storage drive 1414, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive 1414 reads from and/or writes to a removable storage unit 1418 in a well-known manner. Removable storage unit 1418, represents a floppy disk, magnetic tape, optical disk, etc. which is read by and written to by removable storage drive 1414. As will be appreciated, the removable storage unit 1418 includes a computer usable storage medium having stored therein computer software and/or data.

[0069] In alternative embodiments, secondary memory 1410 may include other similar means for allowing computer programs or other instructions to be loaded into computer system 1400. Such means may include, for example, a removable storage unit 1422 and an interface 1420. Examples of such may include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units 1422 and interfaces 1420 which allow software and data to be transferred from the removable storage unit 1422 to computer system 1400.

[0070] Computer system 1400 may also include a communications interface 1424. Communications interface 1424 allows software and data to be transferred between computer system 1400 and external devices. Examples of communications interface 1424 may include a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, etc. Software and data transferred via communications interface 1424 are in the form of signals 1428 which may be electronic, electromagnetic, optical or other signals capable of being received by communications interface 1424. These signals 1428 are provided to communications interface 1424 via a communications path (i.e., channel) 1426. This channel 1426 carries signals 1428 and may be implemented using wire or cable, fiber optics, a

phone line, a cellular phone link, an RF link and other communications channels. In an embodiment of the invention, signals 1428 comprise input values, multiplier X and multiplicand Y. Alternatively, these values can be read from secondary memory 1410.

[0071] In this document, the terms "computer program medium" and "computer usable medium" are used to generally refer to media such as removable storage drive 1414, a hard disk installed in hard disk drive 1412, and signals 1428. These computer program products are means for providing software to computer system 1400.

[0072] Computer programs (also called computer control logic) are stored in main memory 1408 and/or secondary memory 1410. Computer programs may also be received via communications interface 1424. Such computer programs, when executed, enable the computer system 1400 to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable the processor 1404 to perform the features of the present invention. Accordingly, such computer programs represent controllers of the computer system 1400.

[0073] In an embodiment where the invention is implemented using software, the software may be stored in a computer program product and loaded into computer system 1400 using removable storage drive 1414, hard drive 1412 or communications interface 1424. The control logic (software), when executed by the processor 1404, causes the processor 1404 to perform the functions of the invention as described herein.

[0074] In another embodiment, the invention is implemented primarily in hardware using, for example, hardware components such as application specific integrated circuits (ASICs). Implementation of the hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant art(s).

[0075] The present invention is advantageous in a variety of applications. Being able to leverage configuration information from any telephone in a telephone system is especially beneficial to users who need to use a telephone temporarily (such as a worker using a telephone at a temporary workspace) or for several users who share a common pool of telephones (such as workers using desks with telephones on an as needed basis). Users can then customize phone features and leverage this customization at any dynamically configurable telephone in a telephone system. This can improve efficiency and utilization of telephone features, such as, configurable button assignments, voice mail, call forwarding, conferencing, hold, transfer, call waiting and/or any other telephone feature. Embodiments of the invention can cover a variety of platforms and applications.

6. CONCLUSION

[0076] The present invention can be implemented in software, firmware, hardware or any combination thereof. In examples, the present invention can be implemented in control logic in an processing device, including but not limited to, a general purpose computer, specific purpose computer, server, workstation, personal computer (desktop, laptop, or palm top), personal digital assistant (PDA), network appliance, network component (e.g., switch), telephone unit, game console, or a set-top box.

[0077] While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It will be apparent to persons skilled in the relevant art that various changes in form and detail can be made therein without departing from the spirit and scope of the invention.

1. A system for enabling users to dynamically configure telephones in a telephone system, comprising:

a switch that directs telephone calls carried between telephones and at least one trunk, said switch including physical and virtual extensions, wherein each physical extension corresponds to a link to a respective telephone and each virtual extension corresponds to a respective user; and

a dynamic configuration module that associates a physical extension of a telephone with a virtual extension input by a user, whereby features associated with the virtual extension input by the user are delivered to the telephone.

2. The system of claim 1, wherein said switch comprises a private branch exchange.

3. The system of claim 1, wherein said switch comprises a central office switch.

4. The system of claim 1, wherein said dynamic configuration module delivers virtual extension configuration data to the telephone.

5. The system of claim 1, wherein said switch supports the features or services delivered to the telephone.

6. The system of claim 1, further comprising a memory coupled to said switch and said dynamic configuration module.

7. The system of claim 6, wherein said memory stores a first group of data including associated physical extensions, ports and phone features.

8. The system of claim 7, wherein said memory stores a second group of data including virtual extensions, user identifiers, phone features, and mapping information, and

wherein said dynamic configuration module sends mapping information to said memory that associates the physical extension of the telephone with the virtual extension input by the user.

9. The system of claim 8, wherein said phone features include at least one selected from the group of call forwarding, voice mail, button-map assignments, conference calling, call waiting, hold, and transfer.

10. The system of claim 1, wherein the telephone includes an input for toggling on and off dynamic configuration of the telephone by a user, and an indicator for indicating whether dynamic configuration is on or off.

11. The system of claim 10, wherein said input comprises a button on a panel of the telephone, and said indicator comprises a visible indicator on the panel.

12. The system of claim 1, wherein said switch further includes a port for enabling an administrator to provision physical and virtual extensions.

13. The system of claim 1, wherein said switch further includes a network port for coupling a VoIP telephone to said switch.

14. The system of claim 1, wherein said dynamic configuration module checks whether the virtual extension input by the user is valid and a password input by the user is valid

before associating the physical extension of the telephone with the virtual extension input.

15. A system for enabling users to dynamically configure telephones in a telephone system, comprising:

a switch that couples telephones and at least one trunk, said switch including physical extensions, wherein each physical extension corresponds to a link to a respective telephone; and

a dynamic configuration module that associates a physical extension of a telephone with a physical extension input by a user, whereby features associated with the physical extension input by the user are delivered to the telephone.

16. The system of claim 15, wherein said switch comprises at least one of a private branch exchange and a central office switch.

17. The system of claim 15, further comprising a memory coupled to said switch and said dynamic configuration module.

18. The system of claim 17, wherein said memory stores a first group of data including associated physical extensions, ports and phone features.

19. The system of claim 18, wherein said dynamic configuration module swaps port information stored in said memory so as to associate the physical extension of the telephone with the physical extension input by the user.

20. The system of claim 15, wherein said switch further includes a network port for coupling a VoIP telephone to said switch.

21. In a telephone system having a switch that couples telephones and at least one trunk, the switch including physical and virtual extensions, wherein each physical extension corresponds to a link to a respective telephone and each virtual extension corresponds to a respective user, a method for dynamically configuring a first telephone for use by a first user comprising:

receiving a virtual extension input by the first user through the first telephone; and

associating a physical extension of the first telephone with the input virtual extension, whereby features associated with the input virtual extension are delivered to the first telephone.

22. The method of claim 21, further comprising presenting a first prompt at the first telephone that requests input of the virtual extension.

23. The method of claim 22, further comprising presenting a second prompt at the first telephone that requests input of a password associated with the first user.

24. The method of claim 23, further comprising checking whether the virtual extension input and password input are valid prior to said associating step.

25. The method of claim 23, further comprising illuminating an indicator of the first telephone in response to the

physical extension of the first telephone being associated with the input virtual extension.

26. The method of claim 23, further comprising providing phone features associated with the input virtual extension to the first telephone, the provided phone features including at least one phone feature selected from the group of call forwarding, voice mail, button-map assignments, conference calling, call waiting, hold, and transfer.

27. The method of claim 21, further comprising enabling an administrator to provision physical and virtual extensions at the switch.

28. The method of claim 21, further comprising:

receiving an inbound call to the input virtual extension;

looking up a physical port based on the input virtual extension; and

routing the inbound call to the first telephone at the looked up physical port.

29. The method of claim 21, further comprising:

receiving a message relating the input virtual extension; and

processing the message based on stored feature data relating to the input virtual extension;

looking up a physical port associated with the input virtual extension; and

delivering data corresponding to the processed message over the looked up physical port to the first telephone at the looked up physical port.

30. The method of claim 21, further comprising:

receiving at a physical port a signal corresponding to an input entered at the first telephone;

looking up virtual extension feature data associated with the physical data at which the signal was received; and

processing the signal based on the looked up virtual extension feature data.

31. In a telephone system having a switch that couples telephones and at least one trunk, the switch including physical and virtual extensions, wherein each physical extension corresponds to a link to a respective telephone, a method for dynamically configuring a first telephone for use by a first user comprising:

receiving a physical extension input by the first user through the first telephone; and

associating a physical extension of the first telephone with the input physical extension, whereby features associated with the input physical extension are delivered to the first telephone.

* * * * *

ATTORNEY DOCKET NO.
062891.1251

PATENT APPLICATION
USSN 10/824,180

Appendix B: Evidence Appendix

Tab 2 - *Marcus*



US005933488A

United States Patent [19]**Marcus et al.**[11] **Patent Number:** **5,933,488**[45] **Date of Patent:** **Aug. 3, 1999**

[54] **AUTOMATED METHOD AND ARRANGEMENT FOR INTEGRATING A TELEPHONE SYSTEM WITH AN ANNOUNCEMENT SYSTEM**

[75] **Inventors:** **Rodrigo Tyrone Marcus**, San Francisco; **William Joseph Beyda**, Cupertino; **Shmuel Shaffer**, Palo Alto, all of Calif.

[73] **Assignees:** **Siemens Information; Communication Networks, Inc.**, both of Boca Raton, Fla.

[21] **Appl. No.:** **08/844,416**

[22] **Filed:** **Apr. 18, 1997**

[51] **Int. Cl.⁶** **H04M 3/50; H04M 11/02**

[52] **U.S. Cl.** **379/217; 379/88.15; 379/88.16; 379/88.23; 379/164; 379/199; 379/374**

[58] **Field of Search** **379/76, 217, 67.1, 379/88.16, 88.23, 88.24, 88.25, 88.26, 88.15**

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,642,425	2/1987	Guinn, Jr. et al.	455/31.2
4,720,848	1/1988	Akiyama	379/88
4,741,020	4/1988	Deal et al.	379/67.1
4,825,456	4/1989	Rosenberg	455/31.2
4,837,797	6/1989	Freeny, Jr.	379/96
4,922,526	5/1990	Morganstein et al.	379/157
5,131,048	7/1992	Farenelli et al.	381/81
5,140,626	8/1992	Ory et al.	455/31.2
5,541,981	7/1996	Lynn	379/88.25
5,594,784	1/1997	Vellus	379/88
5,604,791	2/1997	Lee	379/67
5,754,627	5/1998	Butler et al.	379/63
5,768,347	6/1998	Beyda	379/67

FOREIGN PATENT DOCUMENTS

0 536 949 A2 4/1993 European Pat. Off. H04M 3/54

0 691 777 A2	1/1996	European Pat. Off.	H04M 3/50
60-51396	3/1985	Japan	H04Q 3/58
04176220	6/1992	Japan	H04B 17/26
5-136884	6/1993	Japan	H04M 3/42
2 251 359	7/1992	United Kingdom	H04Q 7/00

OTHER PUBLICATIONS

Copy of International (PCT) Search Report mailed Jul. 28, 1998.

Primary Examiner—Daniel S. Hunter

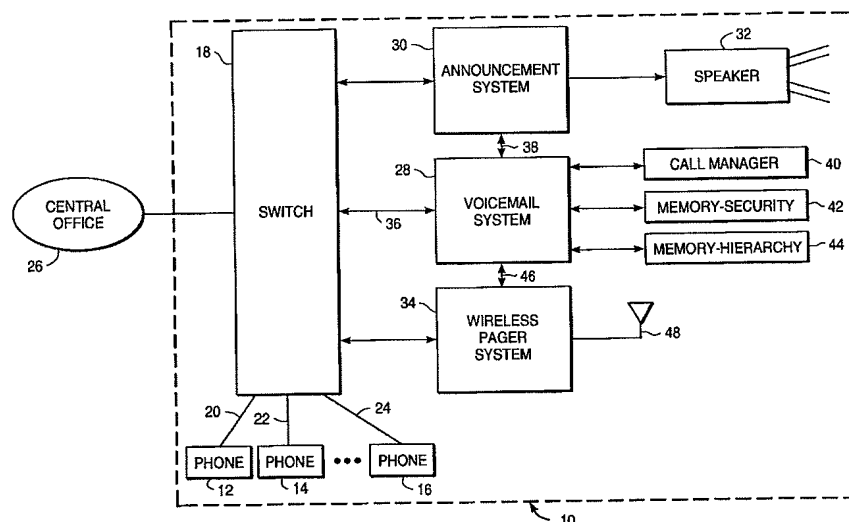
Assistant Examiner—Roland G. Foster

[57]

ABSTRACT

A method and arrangement of utilizing an announcement system to provide paging capability for a facility includes automating the integration of a telephone switch, a voice-mail system, and an announcement system. An unanswered call to a telephone of the facility is forwarded to the voicemail system, establishing a first connection between the telephone of the calling party and the voicemail system. The caller is presented with the option of recording a message or initiating a broadcast of an audible page announcement. If the announcement option is selected, a second connection that is separate from the first connection is formed from the voicemail system to the announcement system. An audible announcement identifies the availability of the call for retrieval by the particular called party. In the preferred embodiment, access to the waiting call is restricted to either or both of verification of a password and verification that the retrieval is from one of a limited number of authorized telephones. Security and privacy are further enhanced by restricting access to the announcement procedure in accordance with the preferred embodiment. Multi-tier call notification may be employed, with differences in the tiers being based either upon regions of broadcast (e.g., localized versus general-facility announcements) or upon modes of paging (e.g., overhead announcements versus a page to a particular remote page device).

16 Claims, 3 Drawing Sheets



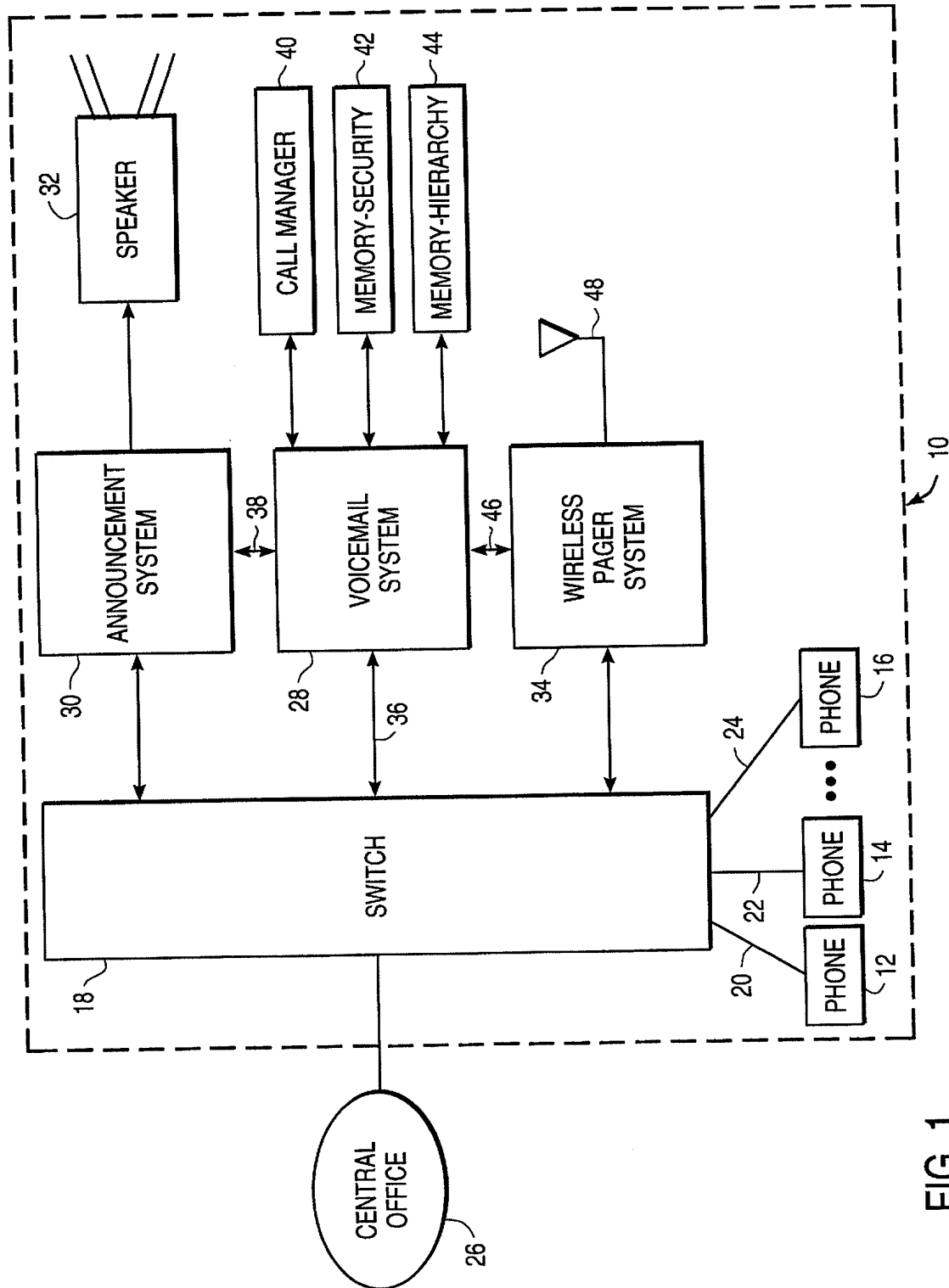


FIG. 1

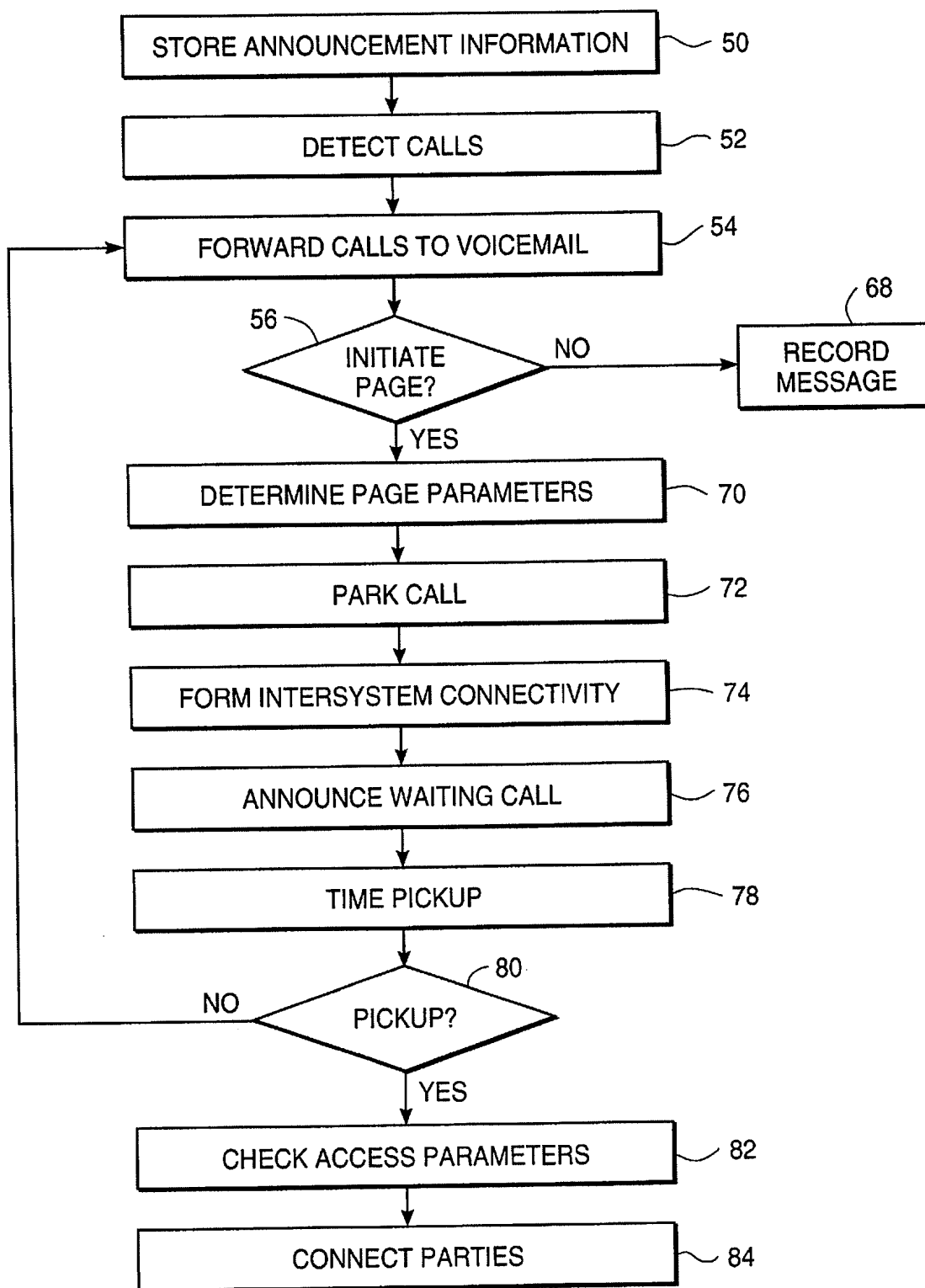


FIG. 2

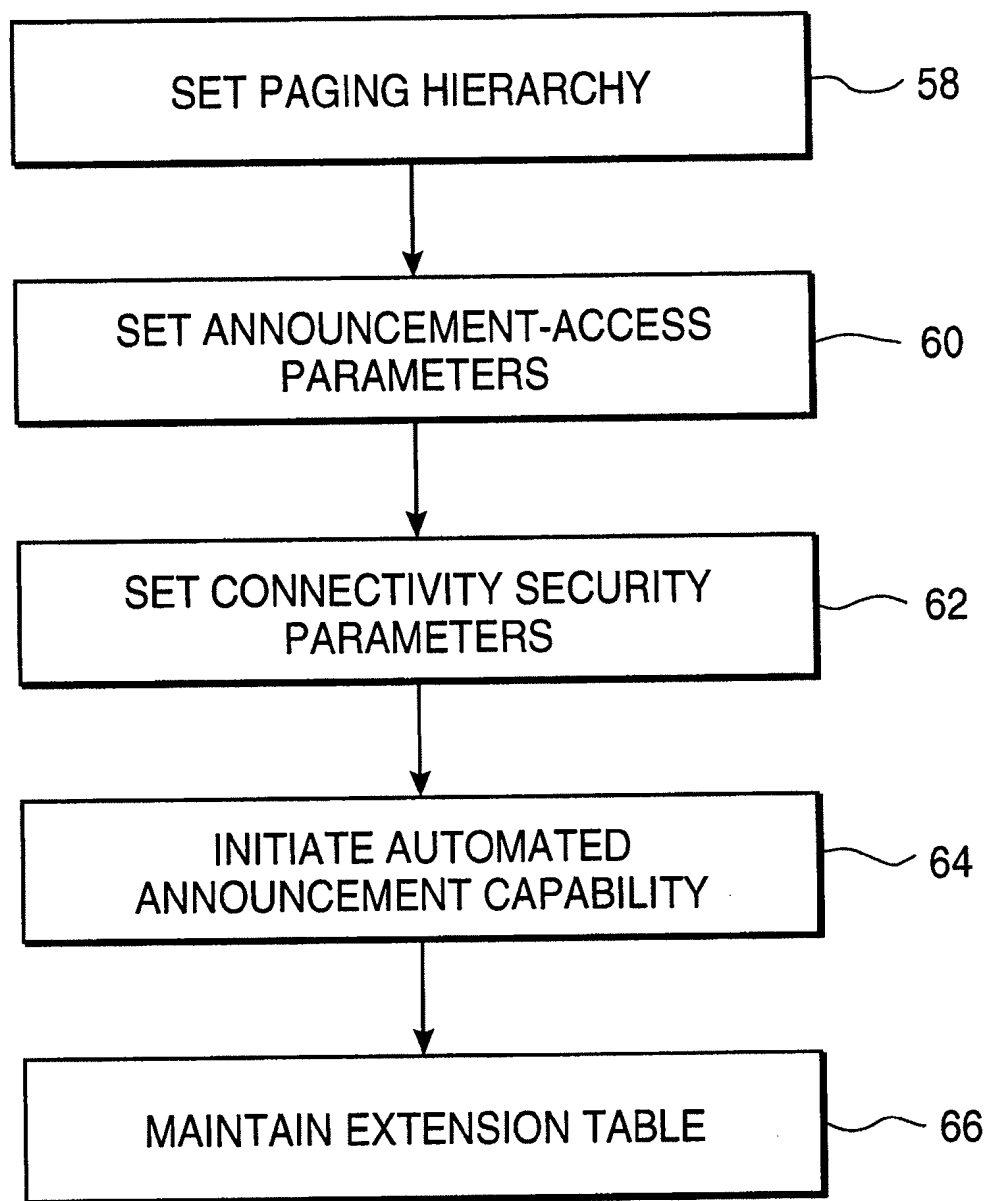


FIG. 3

AUTOMATED METHOD AND ARRANGEMENT FOR INTEGRATING A TELEPHONE SYSTEM WITH AN ANNOUNCEMENT SYSTEM

BACKGROUND OF THE INVENTION

The invention relates generally to methods and systems for notifying a called party that a waiting call is available for pickup and more particularly to automating an announcement system for a facility having multiple telephone units.

DESCRIPTION OF THE RELATED ART

A business or other facility which includes a number of telephones may include a telephone switch that allows direct inward dialing (DID) to a desk of a particular person within the facility. For example, a private branch exchange (PBX) may be used to assign a different extension number for each PBX station line that extends to a telephone. The DID feature accommodates an automatic routing of a call from outside of the facility to the telephone assigned to the called party. Routing of intra-facility calls is also automatic.

If a facility has an announcement system, such as an overhead paging system, the PBX is often configured to route unanswered calls to an operator or receptionist. For example, a telephone in the office of a cardiologist may be allowed to ring four times before the call is routed to an operator employed at a hospital. The operator greets the caller and may offer to page the cardiologist using the overhead system. If the caller requests the page, the operator places the call on hold and issues the page. The call is then monitored to determine whether a person picks up the call. The monitoring may be performed by the operator or by the PBX. Monitoring by the PBX may be implemented by setting a maximum time for which the call can be parked, and generating a ring-back to the operator if the maximum time is exceeded.

One concern with this process is that there are often privacy and security issues. In the hospital example, the parked call may be picked up by someone other than the cardiologist, since the announcement typically identifies both the doctor and the extension on which the call is parked. A calling party may be unaware that he or she is speaking to a person posing as the called party.

Another concern is that the process is labor-intensive. The operator or receptionist must speak with the calling party prior to initiating the announcement, must articulate the announcement, and must handle the ring-back calls if the paged person does not pickup the extension. For the ring-back calls, the operator may be required to take a message or to offer the option of transferring the call to a voicemail system.

Yet another concern in the use of overhead announcement systems to provide call notification involves human inconsistencies. Different persons will have different voice qualities, amplitudes, and clarifies. As a result, there may be some difficulties in understanding pages.

With the exception of the security concern, there have been improvements to the call-notification process. For example, the ring-back situation for unanswered pages may transfer a caller directly to the voicemail box of the called party in another development. U.S. Pat. No. 4,741,020 to Deal et al. describes an overhead announcement system for paging store clerks having specialized knowledge to help customers. For customers on a telephone, clerks are paged using a two-part message in which the first part, typically a

department name, is determined by a button pushed by the store's telephone operator and the second part consists of the telephone line number. A typical message would be "Plumbing, line 2." The use of stored-speech messages to provide the paging standardizes the messages and announcements. However, the security issues remain. Moreover, the announcement process still requires intervention by a telephone operator of the facility.

U.S. Pat. No. 5,131,048 to Farenelli et al. describes an audio distribution system for use in homes. The system controls the broadcast of different types of signals through speakers located in various zones, using a control circuit which responds to inputs of music, intercom, page, and doorbell signals. A telephone handset at the home includes a page button. When the page button is depressed, the telephone handset generates a monaural paging signal. The system includes a page interface circuit that is responsive to the telephone handset to generate a control signal, serve as a power source to the telephone handset, and process an audio signal from the telephone handset. The system operates well within the home environment, but its application to a business facility is limited.

U.S. Pat. No. 5,541,981 to Lynn describes an automated announcement system that allows messages to be played on a public address system and/or displayed on a display device in accordance with a predefined schedule that maintains a minimum interval between messages. The system resolves conflicts between messages to be played at the same time by assigning priorities to the messages. An example of a use of the system is to play recorded announcements that inform customers of special sale events or promotions. In addition to the prerecorded messages, live announcements may be presented. The live announcements may be provided using a connection to a PBX pager output. That is, a call may be made to the PBX by a person who wishes to make an announcement. Preferably, a multi-level password protection scheme is implemented to restrict specific system functions to authorized users. The availability of a live announcement option may require input of a password.

In addition to overhead announcement systems, wireless pagers may be used to notify a party that a call is available. U.S. Pat. No. 4,825,456 to Rosenberg discloses an apparatus for providing call notification via pagers. A number of pager transmitters are connected to telephone extension lines by means of interface circuits. Each interface circuit includes a ring signal detector to detect activation of the associated telephone extension line. The detection triggers a pager transmitter activator, which activates the pager of the appropriate individual.

A similar apparatus is described in U.S. Pat. No. 4,642,425 to Guinn, Jr. et al. An incoming call is placed on hold while the pager of the called party is activated. The called party is then able to pickup the call at a local telephone within the facility. If there is a one-to-one correspondence between the calls and the pagers that are activated in response to the call, the security concerns are alleviated. However, the increase in security comes at a sacrifice of increased system costs, relative to overhead announcement systems.

What is needed is a method and arrangement for utilizing an announcement system to provide paging capability for facilities having a large number of telephones, with the utilization preferably being implemented in a cost-efficient and privacy secure manner.

SUMMARY OF THE INVENTION

A method and arrangement for utilizing an announcement system to provide paging capability includes automating

3

integration of a telephone switch, a voicemail system, and an announcement system of a facility having a number of telephones, with each telephone being associated with a particular user. Unanswered calls are forwarded to the voicemail system, so that a first connection is formed between the voicemail system and a telephone of the calling party. A second connection is then formed between the voicemail system and the announcement system. The second connection is separate from the first connection, preventing the calling party from entering a live announcement. Instead, as an automated response to establishing the second connection, an audible announcement is triggered. The audible announcement identifies the availability of the call for retrieval by a particular called party. In the preferred embodiment, the access to the waiting call is restricted by either or both of verification of a password and verification that the retrieval is from one of a limited number of authorized telephones.

The preferred embodiment further includes parking the call from the calling party prior to triggering the audible announcement, thereby terminating the first connection to the voicemail system and allowing immediate access to the call by the called party. The call is parked on a known telephone line, such as a "dummy extension." The audible announcement includes identifications of both the known telephone line and the called party, i.e., the identified user. A threshold time for pickup is preselected. If the call is not retrieved prior to expiration of the threshold time, the waiting call is returned to the voicemail system.

Still referring to the preferred embodiment, the audible announcement is a first tier of a multi-tier notification scheme. The different tiers may be based upon regions. For example, the first announcement may be a localized announcement within the facility, while the second tier may be audible only within a different localized region or may be a general-facility announcement. Alternatively, the differences in the tiers of the multi-tier notification scheme may be based upon modes of paging the identified party. For example, the first mode is the public announcement system, while a backup mode may be to notify a wireless pager carried by the identified party. Alternative announcement systems are also available. As an example, one audible announcement system may include overhead speakers, while a second announcement system may use the intercom feature of a facility having speaker telephones.

As previously noted, the calling party is connected to the voicemail system prior to any connection to the announcement system. Therefore, the calling party may be presented with certain options. The calling party may select between leaving a voice message and initiating the announcement process. In one embodiment, the access to the announcement process is restricted by input of a password or by recognition that the calling party is at one of a designated number of telephones. If the multi-tier notification scheme is utilized, the transitions from one tier to the next may be strictly time-related or may be implemented at the option of the calling party, e.g., if the use of one tier fails to result in a call pickup, the unanswered call is returned to the voicemail system and the caller is presented with the option of triggering the second tier or leaving a message.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a facility having an automated integration of a switch, a voicemail system, and an announcement system in accordance with the invention.

FIG. 2 is a process flow of steps for utilizing the intersystem arrangement of FIG. 1 for providing paging capability.

4

FIG. 3 is a process flow of initiating the intersystem arrangement of FIG. 1.

DETAILED DESCRIPTION

With reference to FIG. 1, a facility 10 is shown as including a number of telephones 12, 14 and 16. Each telephone is connected to a switch 18 via a different station line 20, 22 and 24. The switch may be a PBX, but this is not critical. While not shown in FIG. 1, the switch preferably includes "dummy extensions," i.e., extensions which are not directly tied to a telephone, allowing a call to be parked for subsequent retrieval.

A call from outside of the facility 10 is routed to the switch 18 via a central office 26, as is well known in the art. Each of the telephones 12, 14 and 16 is assigned to a particular user and has a unique phone number. A direct inward dialing (DID) feature of the switch 18 permits an external caller to reach a particular phone of one of the users. Thus, the called party can be identified by the phone number dialed by the calling party. This phone number is received in the calling information during the setup of the call.

Internal calls from one of the telephones 12, 14 and 16 to another one of the telephones require only the extension number to be dialed by the calling party. Often, this is a two-digit to a five-digit number that is identical to the last several digits that must be dialed by an external caller.

An unanswered internal or external call to one of the telephones 12, 14 and 16 may be directed to a voicemail system 28. The forwarding of a call to a voicemail system is known in the art. The forwarding is typically an automated rerouting by the switch 18. In addition to the switch and the voicemail system, the facility includes an announcement system 30. The announcement system provides audible announcements via at least one speaker 32, and is distinguishable from a wireless pager system 34. The speaker or speakers may be overhead devices or may be contained within each one of the telephones 12, 14 and 16. That is, the telephones may be speaker phones. Other schemes for audibly identifying the availability of calls for pickup by particular parties may also be utilized.

In operation, intersystem cooperation is achieved without requirement of an operator. When a call is unanswered, a first connection 36 is formed between the switch 18 and the voicemail system 28 to connect the calling party with the voicemail system. In the preferred embodiment, the caller is presented with the option to either leave a voice message or initiate an announcement process. For example, the caller may be instructed to press the "1" key of a telephone keypad to select the message option, or to press the "2" key to initiate the announcement process. If the announcement option is selected, a second connection 38 is established. The second connection is illustrated as a direct connection between the voicemail system 28 and the announcement system 30, but typically the connection is formed via the switch 18. A prerecorded message is audibilized via the speaker or speakers 32. The triggered prerecorded message identifies the called party and an extension at which the waiting call is parked. If the switch 18 is utilized to park the call, the first connection 36 may be terminated upon parking the call. On the other hand, if the voicemail system 28 is used to park the call, the first connection 36 remains intact. Particularly in embodiments in which the voicemail system is used to park the call, a call manager 40 is beneficial in handling calls routed to the voicemail system. Two memory modules 42 and 44 are also shown as being connected to the voicemail system. The functions of the memory modules will be described fully below.

Optionally, the wireless pager system 34 may be activated if the paged party does not pickup the waiting call within a preselected period of time. The call is again forwarded to the voicemail system 28, which automatically establishes a connection 46 with the wireless pager system. The information required for paging the called party is transferred from the voicemail system 28 to the wireless pager system. This information may be stored at the second memory module 44. The type of pager system is not critical to the invention. In one embodiment, radio frequency signals that are specific to one remote pager device are transmitted via an antenna 48.

One advantage of the invention is that the automation of the call-notification operations eliminates the need of a full-time operator at the facility 10. Another advantage is that prerecorded paging messages may be used to provide a uniform voice and format and to ensure sufficient clarity, amplitude and quality.

Another important advantage relates to security and will be described fully below. Access to the announcement procedure may be restricted by use of a password or by designating a limited number of authorized telephones. Moreover, access to calls that have been announced may be limited by requiring a password or by designating only certain phones as authorized phones for retrieving a particular parked call. The automated process allows each person who may be the object of a page to be reached within selected paging zones, with password protection.

The call-notification process will be described more fully with reference to FIG. 2. In step 50, announcement information is stored. Typical announcements will include the identification of a person and an extension at which a call is parked. In a hospital setting, an exemplary announcement is "Doctor Green, please pickup extension 2136." The various announcements are stored at the voicemail system 28, but may be stored at the announcement system 30 of FIG. 1.

In step 52, external and internal calls directed to one of the telephones 12, 14 and 16 are detected and routed using conventional techniques. The switch 18 is used to route the calls. PBXs allow a user of one of the telephones 12, 14 and 16 to notify the PBX that the user will be away from the telephone, so that incoming calls are immediately routed to the voicemail system 28. PBXs also allow the phone to ring a set number of times, but then presume that the user is unavailable. In step 54, calls directed to an unavailable user or to a busy phone are forwarded to the voicemail system.

Conventional voicemail systems merely allow a calling party to leave a message in the voicemail box of the called party. However, the preferred embodiment of the invention is one in which the calling party is presented with options. In a personalized greeting, the called user may instruct the caller to press a certain key (e.g., "1") or sequence of keys to leave a message, or to press a different key (e.g., "2") or sequence of keys to initiate a page announcement. The decision step is shown at 56 in FIG. 2.

In addition to recording the page announcement information at step 50 and recording the personalized greeting within the voicemail system 28, there are other setup options and requirements. FIG. 3 illustrates some of the possible setup steps. Firstly, a paging hierarchy may be formed 58. As previously noted, the hierarchy may be based upon area, based upon modes of paging, or both. An area-based hierarchy may have a first level in which the triggered announcement is audible on only one floor of a hospital and may have a second level in which the page announcement is broadcast on all floors of the hospital. A mode-based hierarchy for the facility 10 of FIG. 1 may utilize the announcement system

30 to broadcast the page announcement, with the wireless pager system 34 being used if the page announcement does not result in a call pickup. In the preferred embodiment, the hierarchy is determined on a user-by-user basis. That is, each user is enabled to select a desired hierarchy. However, the selection may be facility-wide. The selected hierarchy is stored in the memory module 44 of FIG. 1.

The setup step 60 in FIG. 3 provides a first level of security. Optionally, only designated individuals are authorized to utilize the announcement system 30. All other callers may merely leave a message at the appropriate voicemail box of the voicemail system 28. In one embodiment, the announcement-access is enabled only if a caller inputs a password, such as a particular sequence of digits. For example, there may be a personal identification number (PIN) that is input using the keypad of a conventional telephone. As an alternative to the password-based access enablement, there may be a limitation to the telephones that may be used to trigger a page announcement. As one example, automated access to the announcement system may be restricted to calls initiated from one internal telephone 12, 14 and 16 to another one of the internal telephones. In another example, caller identification information received as a result of an incoming external call may be used as a basis to determine announcement-access. That is, the announcement-access parameters set in step 60 may be a designation of certain external telephones to which access is to be restricted. Similar to step 58, the parameters of step 60 may be individualized or may be facility-wide. The settings are stored in the memory module 42 of FIG. 1.

In step 62, connectivity-security parameters are set. These parameters may be used to limit the access to a waiting call. Privacy and/or security interests may dictate limitations on the availability of call pickup. As an example, a caller may not recognize a doctor by voice, so that any person within a hospital could attempt to pose as a paged doctor and ask personal questions. The connectivity-security parameters set in step 62 may require a person attempting to retrieve a call to enter a password (e.g., a PIN) before the parties are connected. In another embodiment, these retrieval limitations are related to designating certain internal phones 12, 14 and 16 as authorized phones. A particular user may designate a telephone in a backup office of the user as the only authorized phone for retrieving calls that are the subjects of page announcements identifying the user. The retrieval limitations are preferably selected on a user-by-user basis. The parameters are stored in the memory module 42 of FIG. 1.

The automated announcement capability is then initiated at step 64. As will be explained more fully below, the system monitors available extensions for parking the calls that are the subjects of page announcements. Step 66 is included to identify the process of maintaining the extension table.

Returning to FIG. 2, the decision step 56 is determined at a first level by the announcement-access parameters selected in step 60 of FIG. 3 and stored in the memory module 42 of FIG. 1. At a second level, the calling party selects between recording a message and triggering a broadcast of a page announcement. Step 68 of recording a message is implemented using conventional techniques of voicemail systems. On the other hand, if the calling party is authorized and elects the announcement option, the relevant page parameters are determined at step 70. The selected paging hierarchy of the memory module 44 may dictate that the page announcement is to be made only within a localized region. In some embodiments, the calling party is presented with options relating to the area in which the page announcement is to be broadcast.

The voicemail system 28 then parks the call at step 72. A limited number of "dummy extensions" may be configured in the switch 18. Each of the dummy extensions is capable of holding a waiting call. The voicemail system may implement this step by sending a park command to the switch 18. As an example, the park command may be PARK 2136, where PARK is the system PARK command or button and where 2136 is one of the dummy extensions. In order for the voicemail system to monitor the extensions, the extensions must be preconfigured within the voicemail system. As noted with reference to step 66 in FIG. 3, the voicemail system may use an extension table to track the hold-and-retrieval process. While the call is parked at a dummy extension, the caller will hear ring-back tone or on-hold music, depending upon the configuration of the switch 18.

The voicemail system 28 is then able to release the line, since the call is parked on the dummy extension. The voicemail system then goes off-hook on an available channel/port to form the second connection 38 between the voicemail system 28 and the announcement system 30. This intersystem connectivity is shown at step 74 in FIG. 2. While the connectivity is illustrated as a direct connection in FIG. 1, typically the connectivity requires the operation of the switch 18. The off-hook condition of the available channel/port relates to a connection to the switch 18. The voicemail system dials P, where P is the system paging access command. This may be controlled by the call manager 40. The call manager then causes the appropriate announcement information to be played, e.g., "Doctor Green, please pickup extension 2136." The announcement is shown at step 76 in FIG. 2. The interconnection between the voicemail system 28 and the announcement system 30 may then be terminated.

Extension 2136 is marked in the extension table as being occupied by the waiting call. The next caller requesting an announcement may be parked at dummy extension 2137, until all of the available dummy extensions are utilized. If no dummy extension is available for a call, the voicemail system can transfer the next caller to an available operator.

In step 78, the time required for a paged party to retrieve a call is monitored. Preferably, a recall time is established by configuring the voicemail system 28 or the switch 18. If during the decision step 80 the parked call is retrieved, any relevant access parameters are checked at step 82. This includes enforcing the connectivity-security parameters set at step 62 in FIG. 3. The security check is intended to preserve the privacy of the calling party and/or to reduce the susceptibility of the intersystem process to unauthorized dissemination of information. Provided that the person attempting to retrieve the call is authorized, the parties are connected at step 84.

The voicemail system 28 preferably includes a parameter that defines a wait-time between parking calls on an individual dummy extension. Immediately after a call is parked on the extension, the voicemail system starts the timer. After the time expires, the voicemail system assumes that the extension is again available for a parked call. This parameter should be a few seconds longer than the recall time described with reference to the step 78 of timing the call retrieval.

The call retrieval time relevant to steps 78 and 80 is configured in the switch 18 as the maximum time that a call remains parked on an extension before a recall is initiated. In the embodiment of FIG. 2, if the call is not retrieved before expiration of the designated time, the call is again forwarded to the voicemail system at step 54. The voicemail system typically receives some identifying information in

the signaling channel, if one exists (e.g., an ISDN environment). At the least, caller identification information is forwarded to the voicemail system with an indication that the call is a returning one from the PARK condition. As a result, the voicemail system plays the appropriate prompt, such as "Doctor Green did not answer the page." In the preferred embodiment, the prompt also includes options. The caller may again be presented with the option of recording a message at step 68. In a multi-tier scheme, the caller may be presented with the option of implementing a second tier of call notification. As previously noted, the second tier may be a difference with respect to the area in which an audible announcement is broadcast, or may be a switch to a different system. In FIG. 1, the call manager 40 may initiate the connection 46 to activate the remote pager of the called party. While the connection 46 is shown as the direct connection between the voicemail system 28 and the wireless pager system 34, typically the interaction between the two systems is achieved utilizing the switch 18. For example, the voicemail system may go off-hook on an available channel/port to the switch 18 and dial the appropriate access command for activating the wireless pager system. Some wireless pager systems allow the display of numeric codes on the remote pager devices. In such an embodiment, the voicemail system may be used to outpulse the required dual tone multi-frequency (DTMF) tones to notify the user that he or she can call in and pickup the parked call. Pagers that include alphanumeric displays are contacted by the voicemail system using the appropriate protocol for the particular system.

Remote pager devices may play short voice messages. In such a situation, the voicemail system 28 of FIG. 1 may contact the pager system 34 and play the same message that was sent to the announcement system 30 for broadcast. A special external pickup feature may be implemented in the switch 18 to allow a user to call in from an external phone in order to retrieve the parked call. As an alternative, the paged external caller may contact an operator at the facility 10 and request that the operator establish the connectivity between the parties.

In embodiments that include the multi-tier notification scheme, the appropriate sequence of steps that follow the decision step 56 in FIG. 2 are followed a second time. Thus, the parked call is either retrieved prior to the expiration of the time limit of step 78 and the parties are connected at step 84, or the call is returned to the voicemail system from the decision step 80. If all of the tiers are implemented without success, the calling party is informed that the only remaining option is recording a message at step 68.

As an alternative to the embodiment described above, the unanswered call may be parked logically or physically in the voicemail system 28. The user who is paged would then access the call by logging onto the voicemail system. As another alternative to the arrangement described above, the central office 26 of FIG. 1 may perform the functions of the switch 18. That is, an on-site switch is not critical.

While the invention has been described and illustrated as including a "parking lot" of dummy extensions, this is not critical. Switches having built-in parking areas are known, so that there is no need for the preconfiguration of dummy extensions. A system parking area is sometimes referred to as a system park, a system orbit, or an array of system spaces.

What is claimed is:

1. A method of utilizing an announcement system to provide paging capability for calls forwarded to a voicemail system of a facility having a plurality of telephones, each

telephone being associated with a particular user, said method comprising steps of:

directing a call received at said facility to a specific telephone associated with an identified user, said call being intended for said identified user;
 transferring said call to said voicemail system as an automated response to determining that said specific telephone has not been answered within a predetermined time interval, including establishing a first connection between said voicemail system and a telephone of a calling party, said voicemail system being capable of storing a voice message from said calling party for subsequent retrieval by said identified user;
 as a first step establishing a second connection between said voicemail system and said announcement system, said second connection being separate from said first connection; and
 parking said call on a known telephone line to terminate the first connection;
 establishing a second connection between said voicemail system and said announcement system, said second connection being separate from said first connection; and
 as an automated response to establishing said second connection, triggering an audible announcement of availability of said call for pickup by said identified user;
 wherein the audible announcement is capable of verbally identifying said known telephone line and said identified user;
 wherein at least one of the steps of triggering said audible announcement and establishing connectivity is selectively implemented to restrict access to either or both of triggering said areawide audible announcement and said pickup of parked call at said telephone line; and
 as a second and following steps triggering said audible announcements in tiers of a multi-tier call-notification scheme in which differences in tiers are based upon at least one of regions of said facility in which said audible announcement is broadcast and modes for identifying said identified user.

2. The method of claim 1 further comprising a step of selecting a threshold time for pickup of said call by said identified user following said audible announcement, wherein said call is returned to said voicemail system for recording a voice message from said calling party upon expiration of said threshold time.

3. The method of claim 1 further comprising a step of routing said call such that said call is accessible by said identified user via a third connection, wherein establishing said third connection includes imposing at least one access restriction.

4. The method of claim 3 wherein said step that includes imposing said at least one access restriction further includes requiring input of a predetermined password prior to establishing said third connection.

5. The method of claim 3 wherein said step that includes imposing said at least one access restriction further includes storing identifications of a limited number of said telephones through which calls to said identified user can be accessed.

6. The method of claim 1 wherein said step of establishing said second connection is executed selectively to accommodate restrictive access to said announcement system based upon identification of said calling party.

7. The method of claim 6 wherein said step of accommodating restrictive access includes requiring said calling

party to input a password as a condition to establishing said second connection.

8. The method of claim 6 wherein said step of accommodating restrictive access includes storing caller identification information indicative of selected telephones and further includes comparing said stored caller identification information to call data received in establishing said first connection, said call data being indicative of said telephone of said calling party.

9. A call-notification method for a facility having a plurality of telephones and an announcement system; with each telephone being associated with a particular user, said method comprising steps of:

determining a presence of a call directed to a first telephone that is associated with a first user;

presenting options to a calling party of said call to record a voicemail message or to trigger an audible announcement, said options being presented when said call is unanswered at said first telephone;

parking said call;

in response to selection of said option to trigger said audible announcement, as a first step triggering an areawide audible announcement on the said announcement system that verbally identifies said first user as having an available call; and

establishing connectivity of said call to one of said telephones in response to pickup of said parked call; and

as a second and following steps triggering said audible announcements in tiers of a multi-tier call-notification scheme in which differences in tiers are based upon at least one of regions of said facility in which said audible announcement is broadcast and modes for identifying said identified user;

wherein at least one of steps of presenting said option to trigger said audible announcement and establishing connectivity is selectively implemented to restrict access to either or both of triggering said areawide audible announcement and said pickup of said parked call.

10. The method of claim 9 wherein said step of presenting said options includes connecting said call to a voicemail system and providing a voicemail prompt that presents said options.

11. The method of claim 9 wherein said selective implementation includes storing a password for each user such that there is a one-to-one correspondence between passwords and said users.

12. The method of claim 9 wherein said selective implementation includes storing identifications of a limited number of telephones to which said access is extended.

13. An intersystem arrangement for a facility having a plurality of telephones comprising:

a switch for handling connections for calls directed to and from said telephones;

a voicemail system connected to said switch to receive calls which are unanswered at telephones to which said calls are directed, said voicemail system having established voice prompts, including a first voice prompt indicative of an option to trigger an areawide audible announcement of a called party, park the said received call at a known telephone line on the said switch, said voice prompts including a second voice prompt indicative of an option to record a message from a caller;

an announcement system responsive to a selection of said option of said first voice prompt, said announcement

11

system being connected to speaker means for audibilizing said areawide audible announcement that verbally identifies said known telephone line and identified user in a multi-tier call-notification scheme in which differences in tiers are based upon at least one of regions of said facility in which said audible announcement is broadcast and modes for identifying said identified user; and

security means for restricting access to call announcement and pickup capability provided by interconnection of said voicemail system with said switch and said announcement system.

12

14. The intersystem arrangement of claim 13 wherein said security means includes memory containing passwords having a one-to-one correspondence with authorized users.

15. The intersystem arrangement of claim 13 wherein said security means includes memory containing identifications of telephones for which said access is authorized.

16. The intersystem arrangement of claim 13 wherein said security means relates to restricting access to retrieval of calls which are unanswered and which are identified in one of said areawide audible announcements.

* * * * *

ATTORNEY DOCKET NO.
062891.1251

PATENT APPLICATION
USSN 10/824,180

Appendix B: Evidence Appendix

Tab 3 - *Baffles*



US006292792B1

(12) **United States Patent**
Baffes et al.

(10) **Patent No.: US 6,292,792 B1**
(45) **Date of Patent: Sep. 18, 2001**

(54) **SYSTEM AND METHOD FOR DYNAMIC KNOWLEDGE GENERATION AND DISTRIBUTION**

(75) Inventors: **Paul T. Baffes; Siddarth Subramanian**, both of Austin, TX (US); **Shane V. Nugent**, Barrington, IL (US)

(73) Assignee: **Intelligent Learning Systems, Inc.**, Austin, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/277,861

(22) Filed: **Mar. 26, 1999**

(51) Int. Cl.⁷ **G06F 15/18**

(52) U.S. Cl. **706/45**

(58) Field of Search 706/45, 46; 434/350; 707/513; 345/333-339, 356

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,311,422	5/1994	Loftin et al.	364/401
5,727,950	3/1998	Cook et al.	434/350
5,761,683	6/1998	Logan et al.	707/513
5,778,368	7/1998	Hogan et al.	707/10
6,032,141 *	2/2000	O'Connor et al.	706/45
6,134,539 *	10/2000	O'Connor et al.	706/45
6,184,885 *	2/2001	DeStefano	345/356

OTHER PUBLICATIONS

Microsoft® Corp.: "Getting Results with Microsoft® Office 97", 1996, pp. 452-469, 680.

ForeFront, Inc.: "ForeHelp User's Manual, Help-Authorizing System For Microsoft Windows", 1994, Version 1.0, pp. 31, 111, 232.

Artificial Intelligence and Tutoring Systems, by Etienne Wenger, Morgan Kaufmann Publishers, Inc., 1987.

Automatic Student Modeling and Bug Library Construction Using Theory Refinement, by Paul Thomas Baffes, 1994.

Extending Domain Theories: Two Case Studies in Student Modeling, by D. Sleeman, Haym Hirsh, Ian Ellery, In-Yung Kim, Kluwer Academic Publishers, 1990.

* cited by examiner

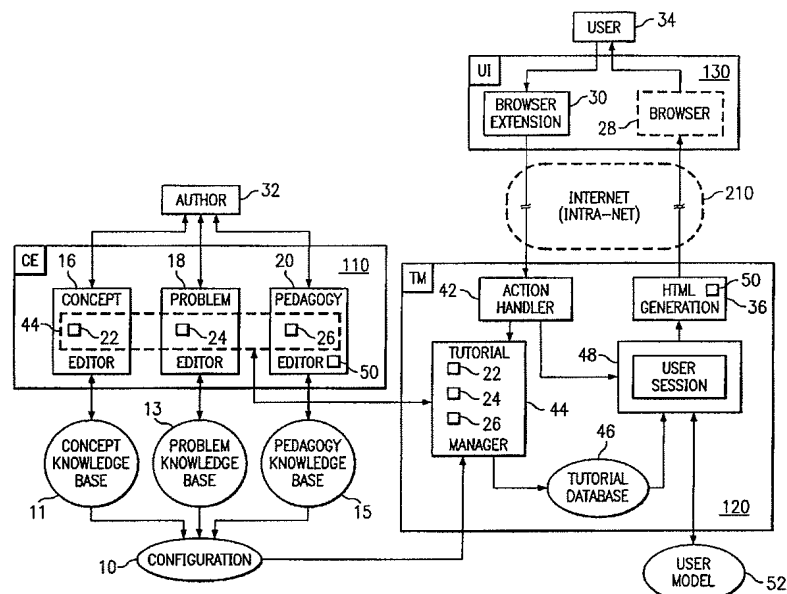
Primary Examiner—Kakali Chaki

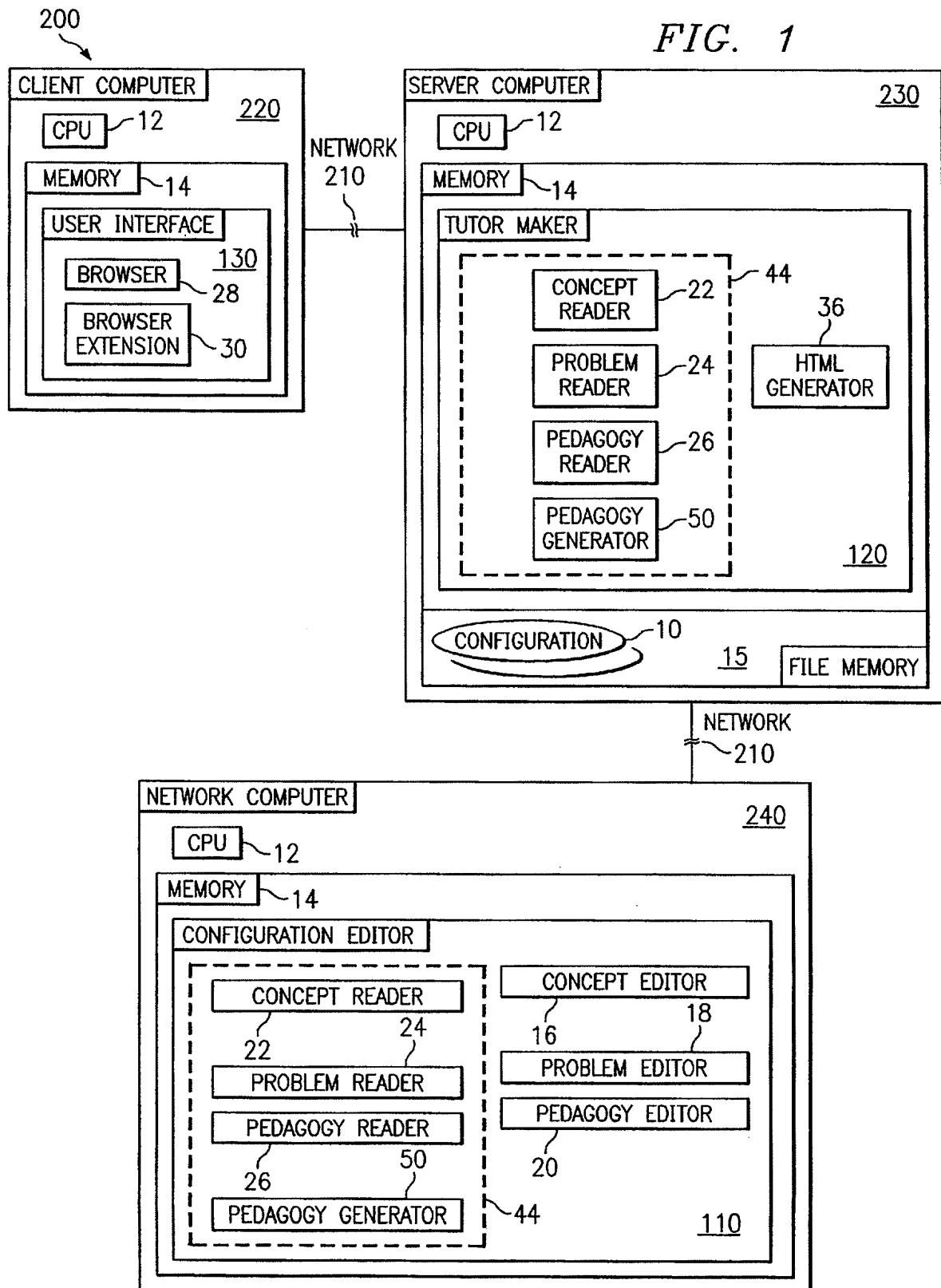
(74) Attorney, Agent, or Firm—Gray Cary Ware and Freidenrich

(57) **ABSTRACT**

A system and method that includes the use of a knowledge delivery computer program stored in computer-readable form on a tangible storage medium for delivering concepts from a configuration data base to a user. An author can create the configuration data base using the knowledge delivery computer program so that the configuration includes a plurality of concepts and, optionally, a plurality of problems associated with the concepts. As the author creates the concepts within the configuration, the knowledge delivery software allows the author to create a taxonomy that defines the relationships between the concepts. The knowledge delivery software can then automatically generate a pedagogy for the configuration based on the configuration taxonomy that defines how the concepts will be delivered to the user. The knowledge delivery software can facilitate the delivery of the content within the configuration to a user according to the pedagogy.

80 Claims, 35 Drawing Sheets





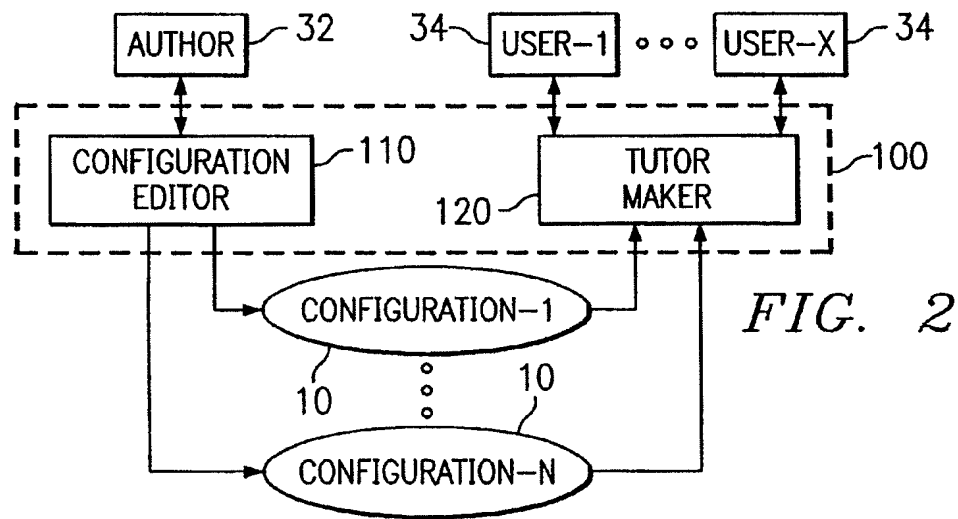


FIG. 2

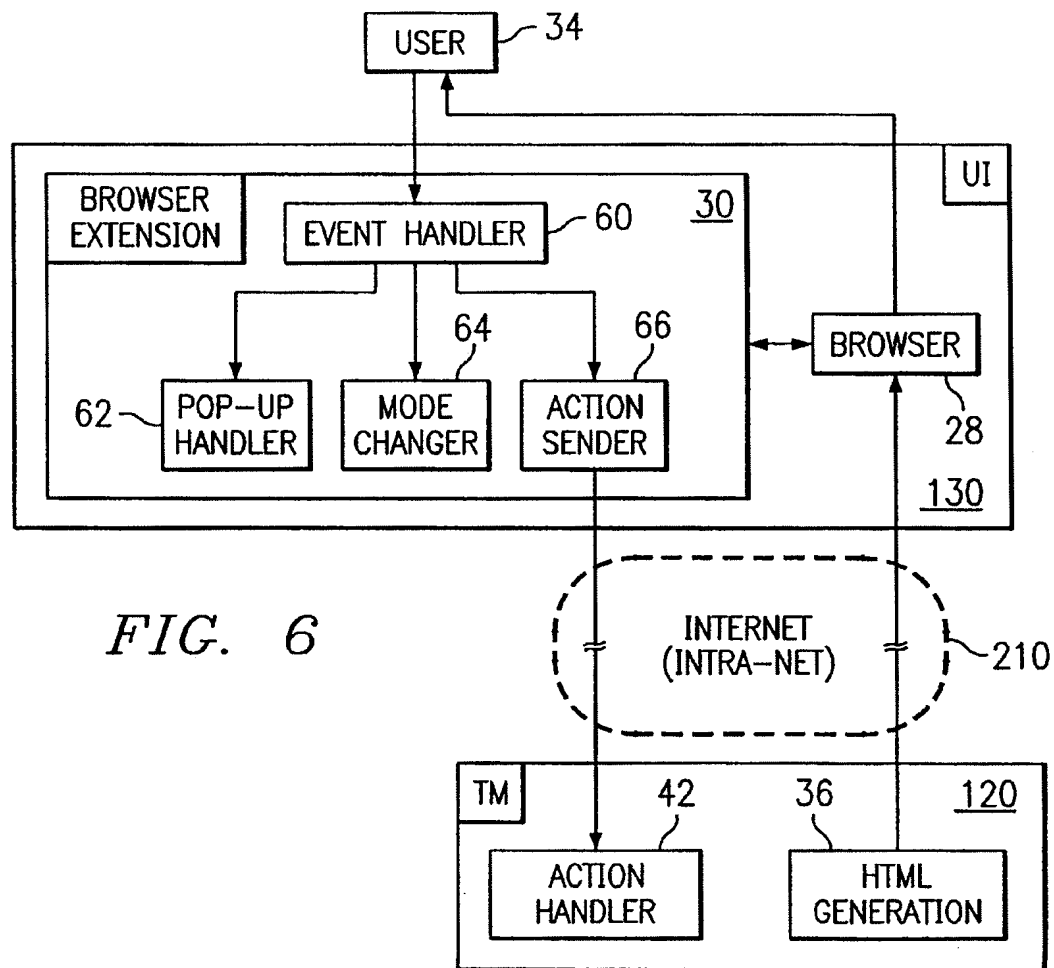
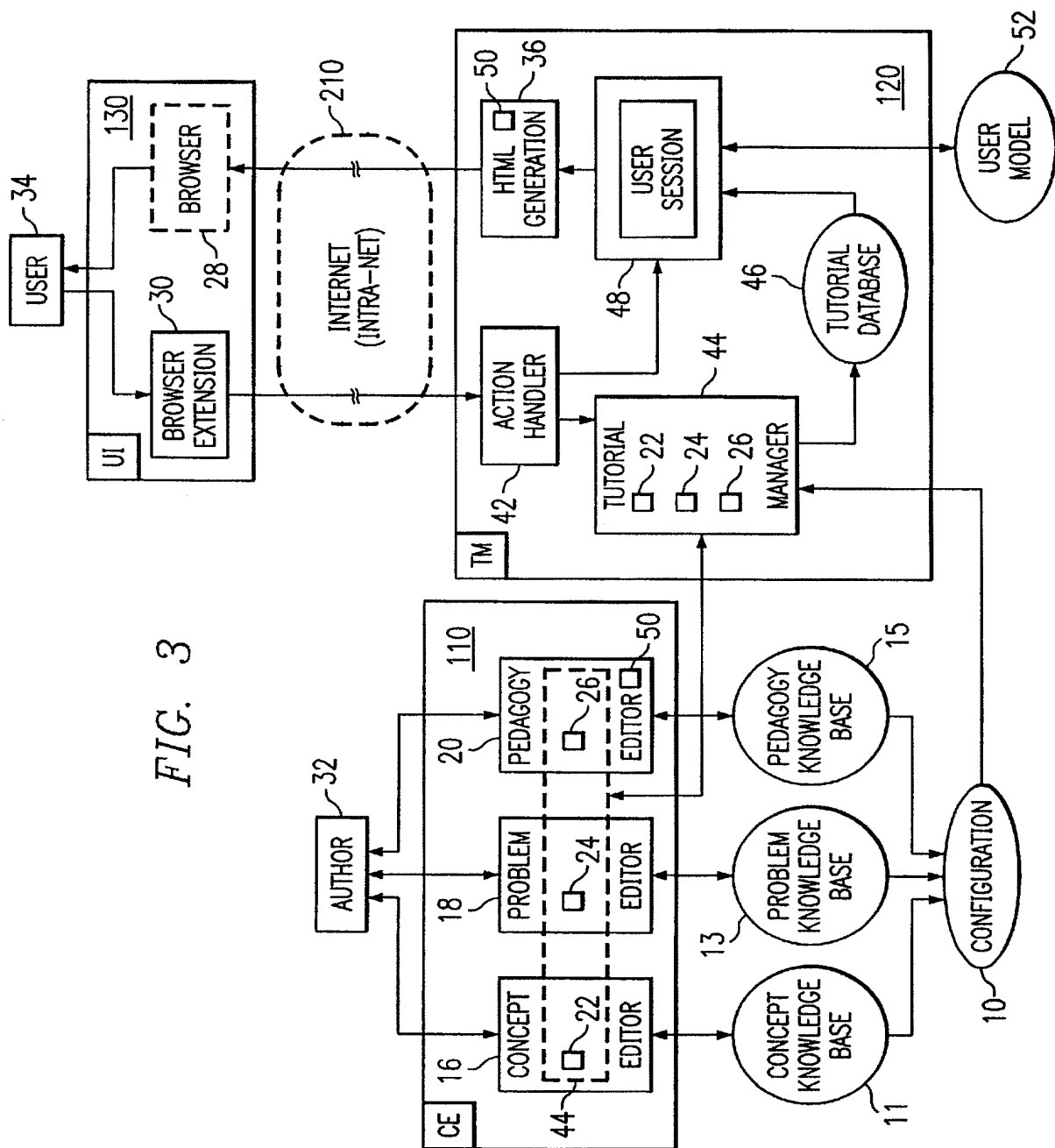


FIG. 6



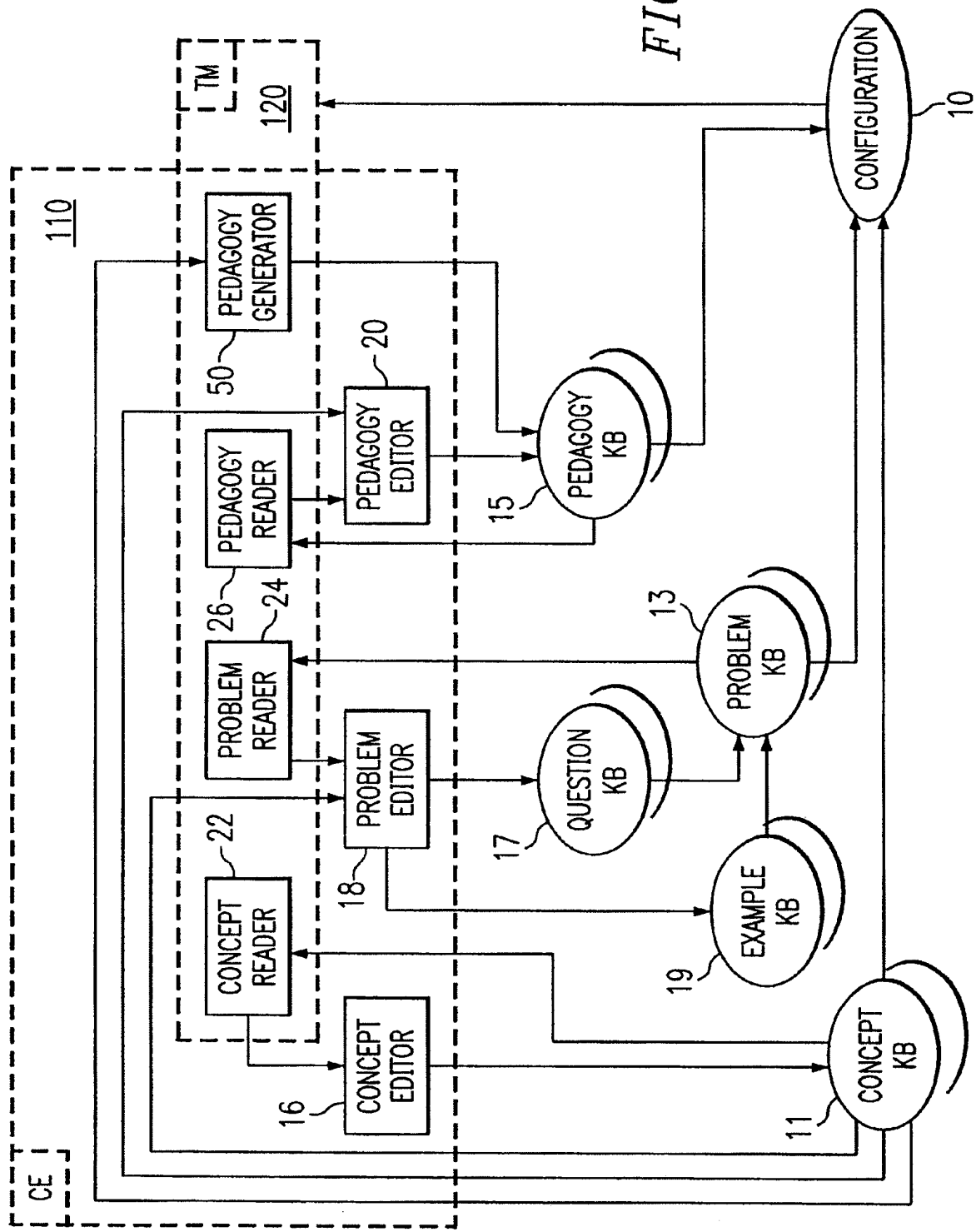


FIG. 4

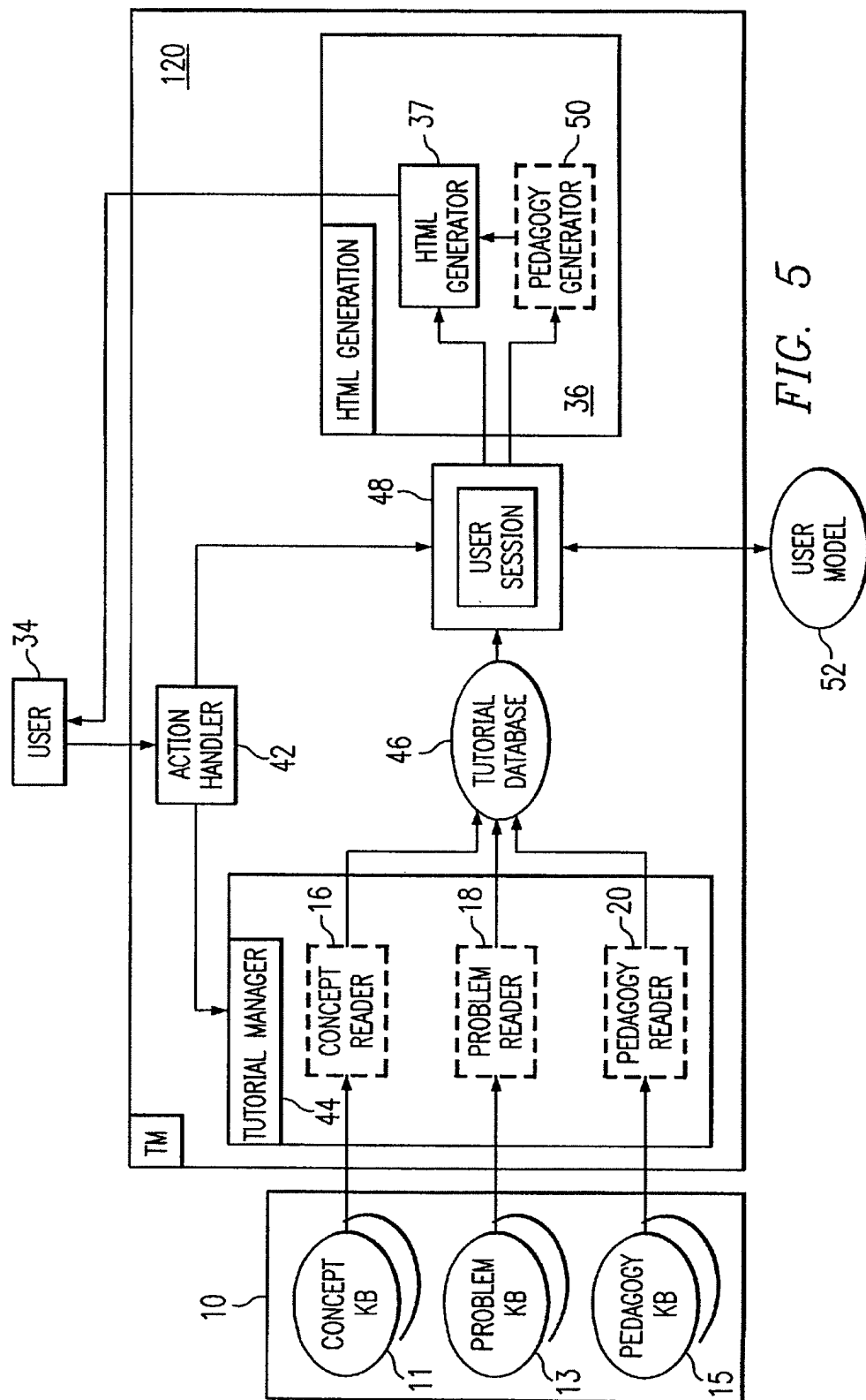
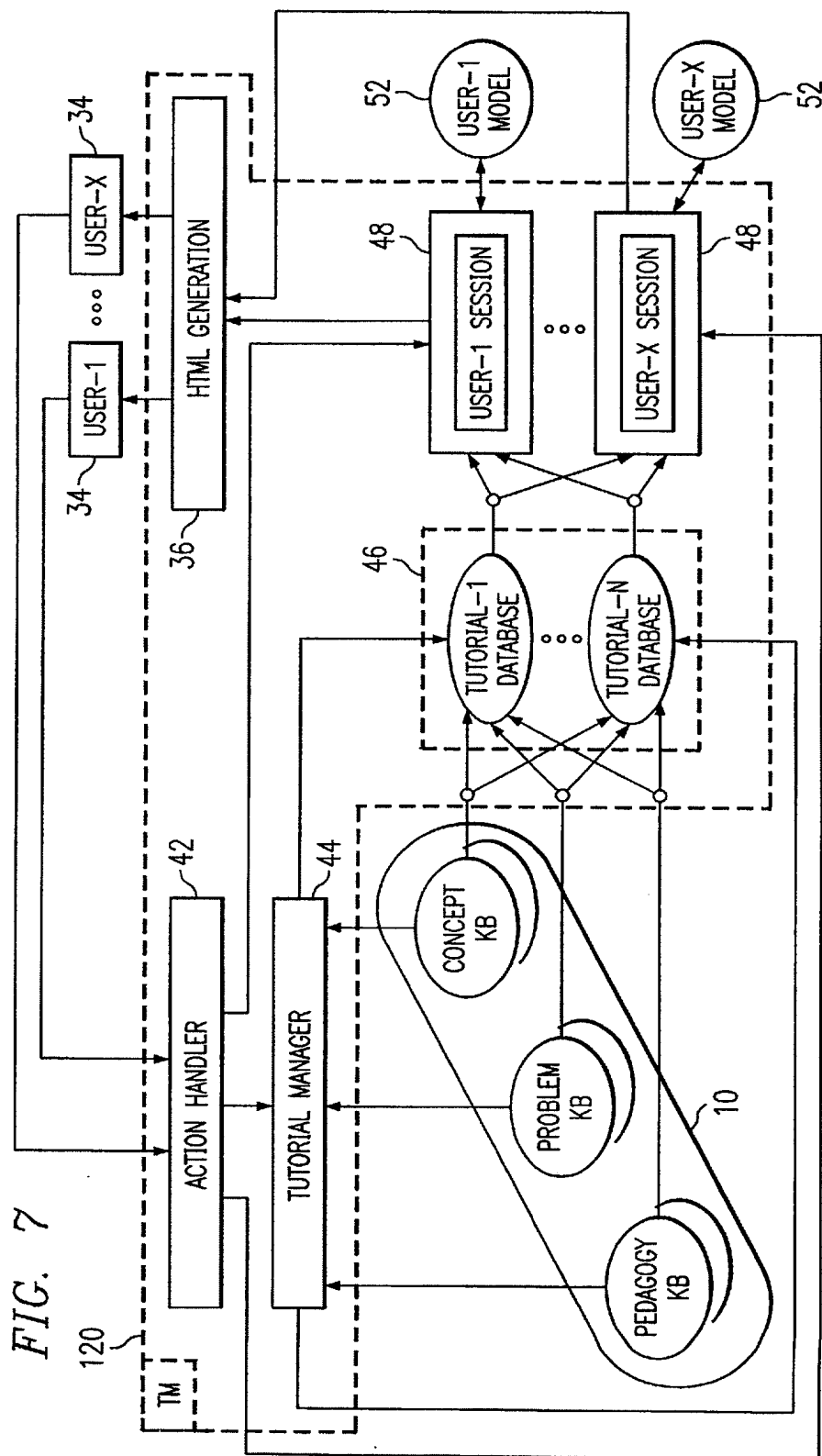
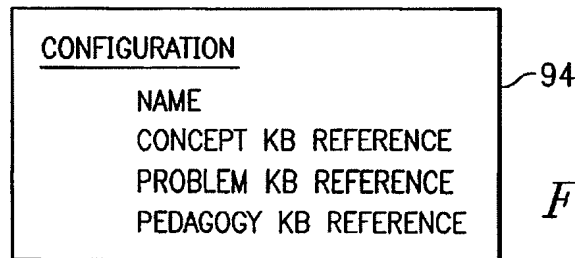
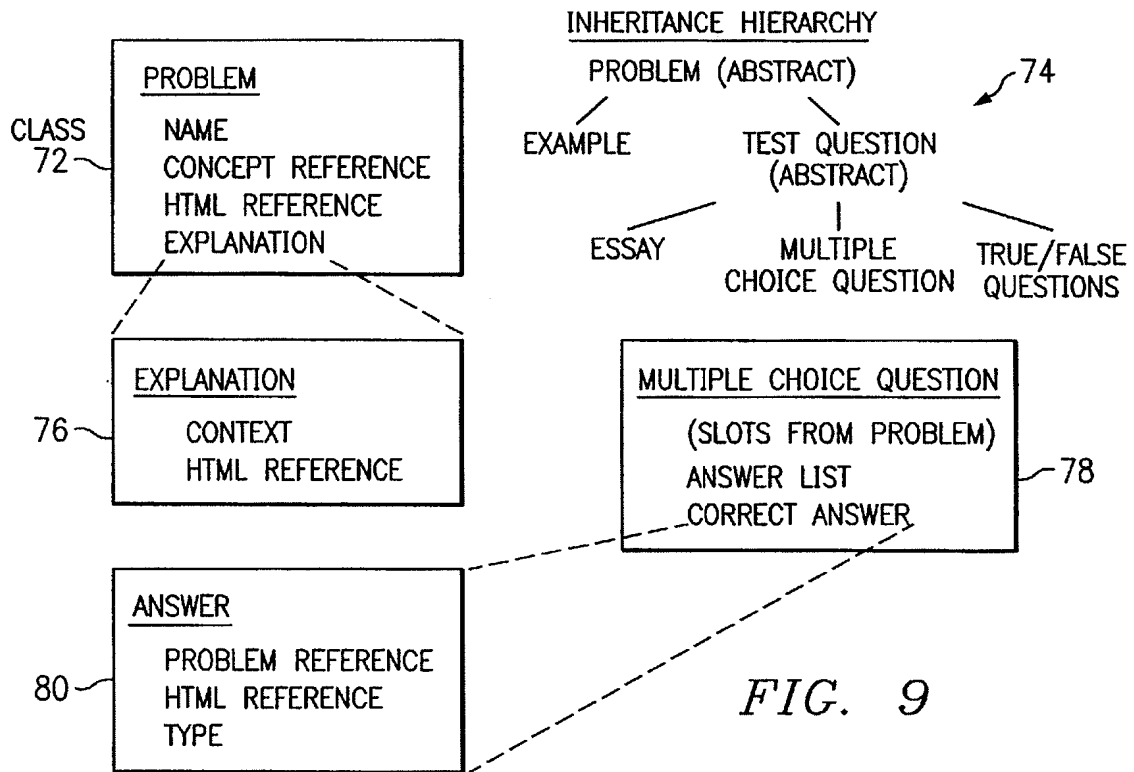
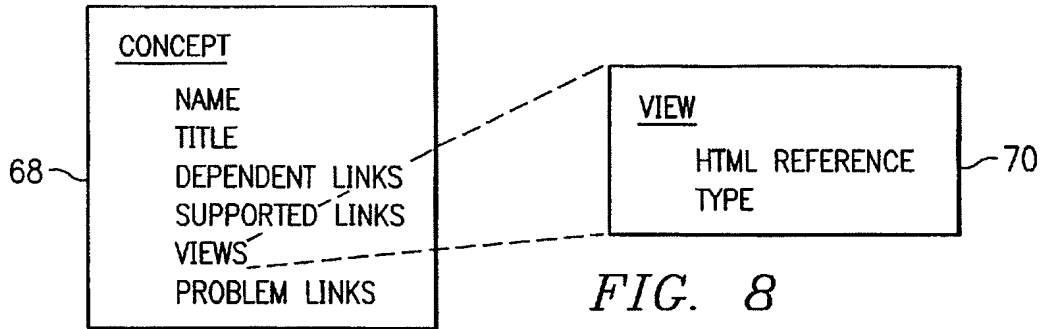


FIG. 5





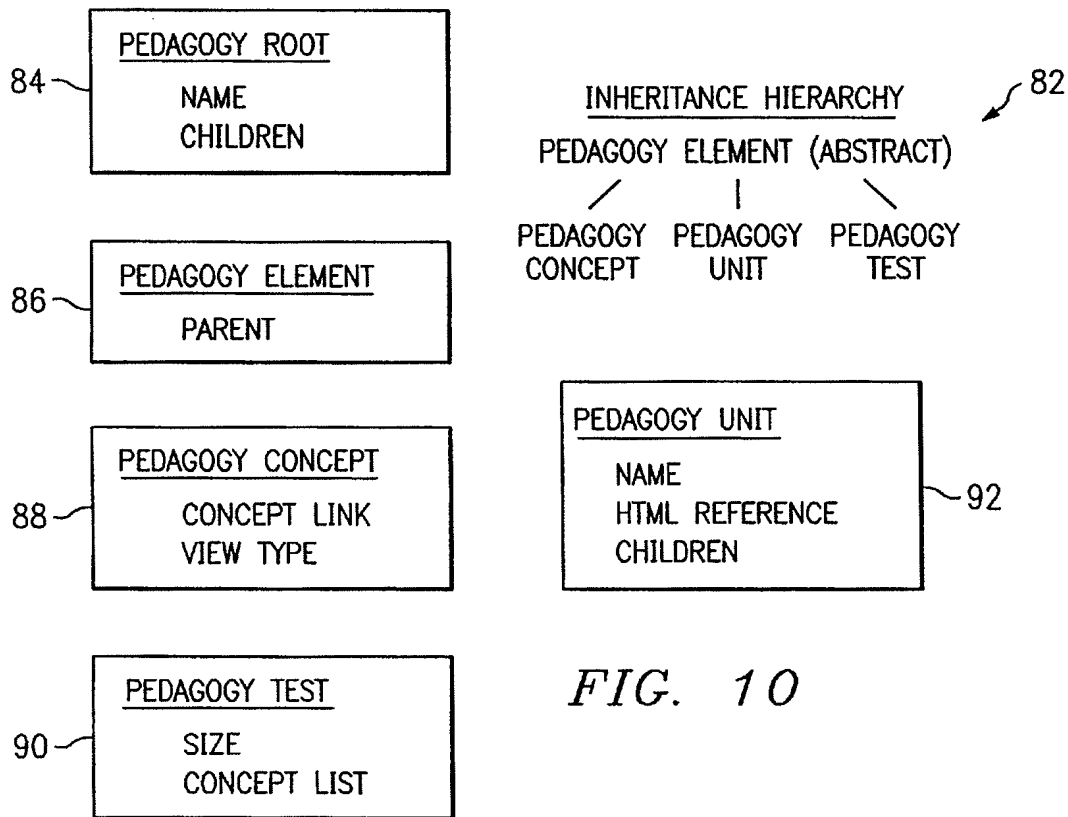


FIG. 10

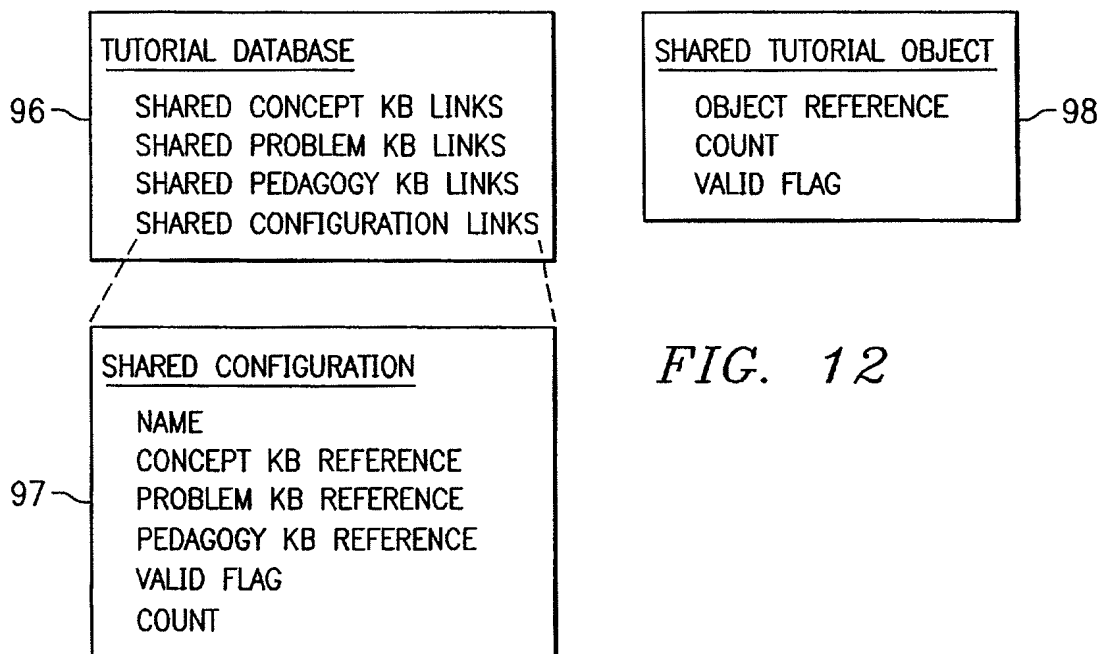


FIG. 12

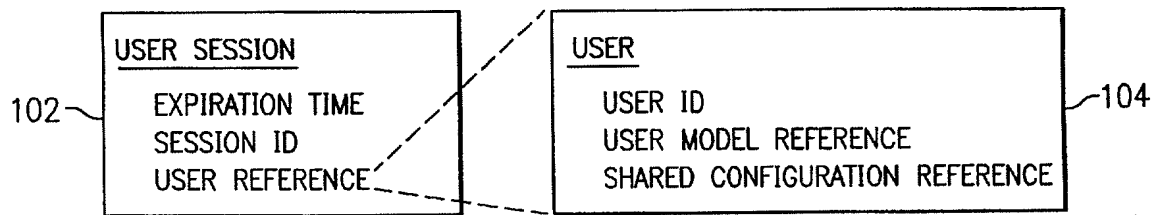


FIG. 13

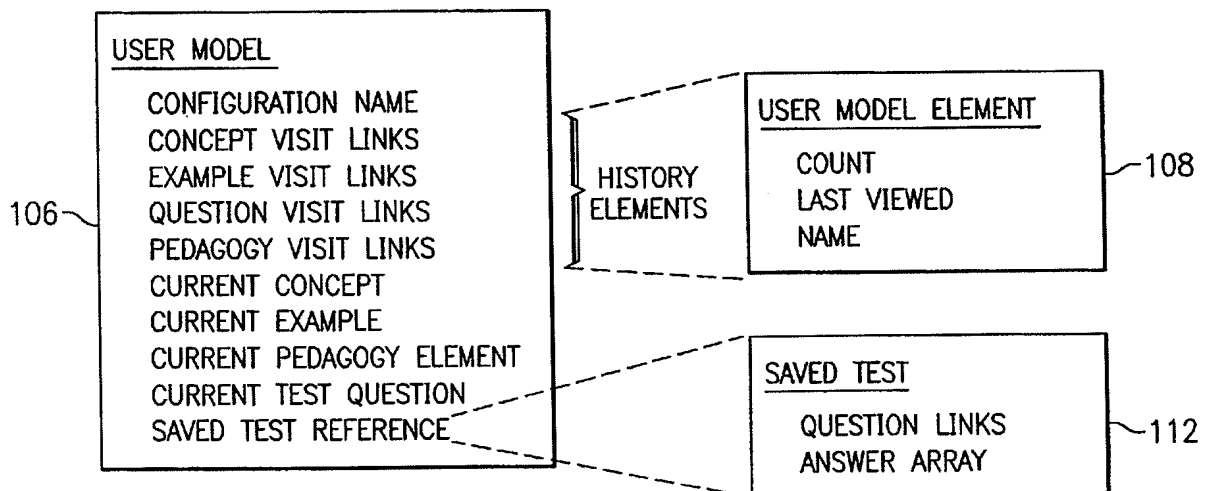


FIG. 14

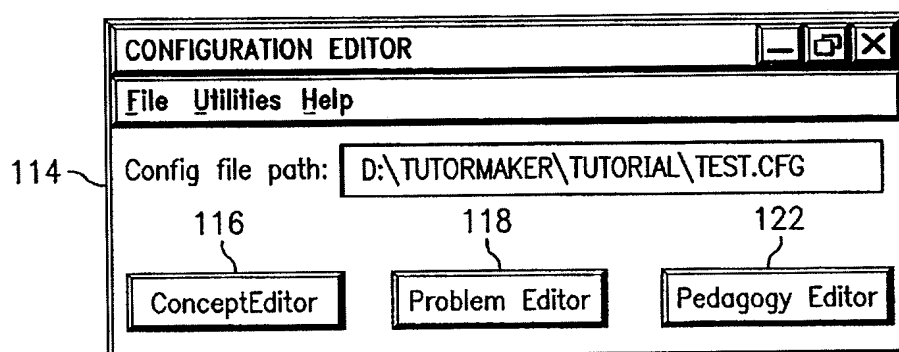


FIG. 15

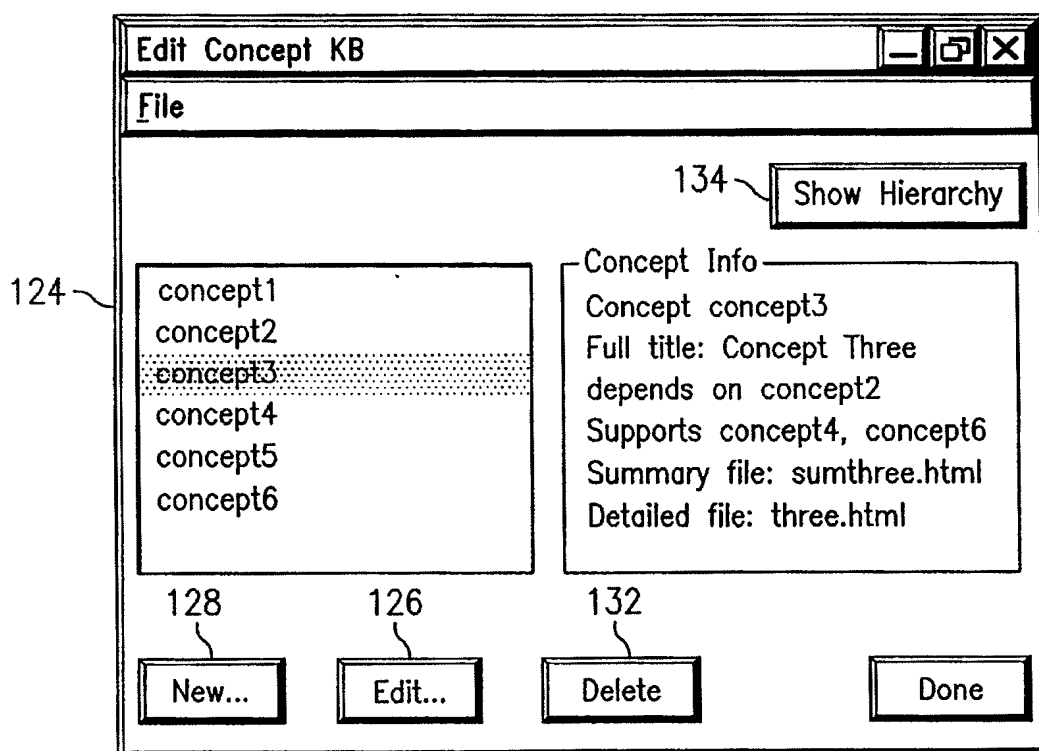


FIG. 16

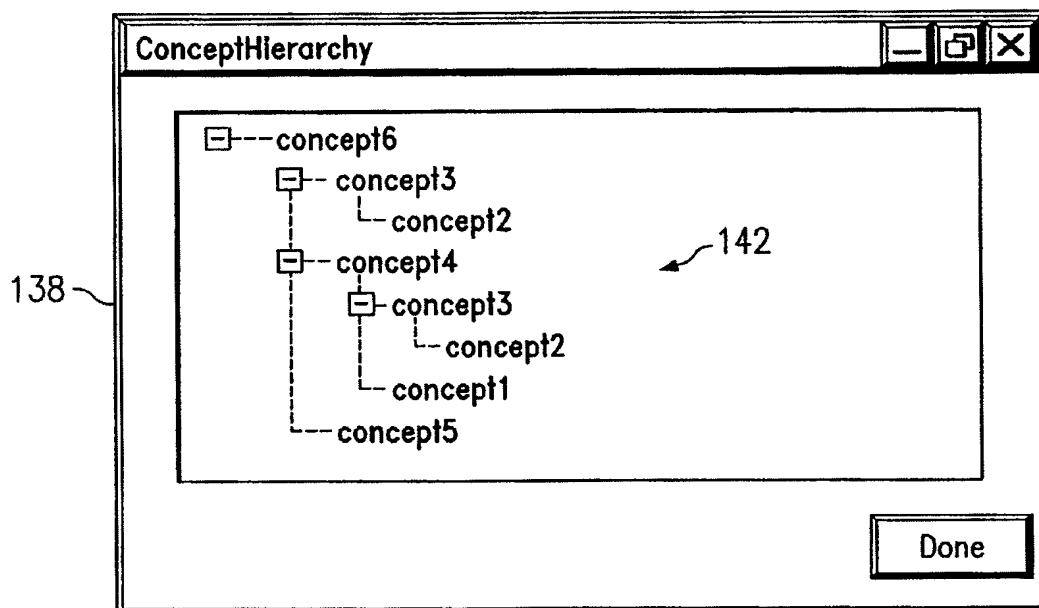


FIG. 18

OneConceptDialog

Concept Identifier: 142

Enter or select the names of any dependent concepts below:

--None--

--None--

--None--

--None--

--None--

--None--

Full name of Concept: 144

Summary HTML 146

Detailed HTML 148

152

136

FIG. 17

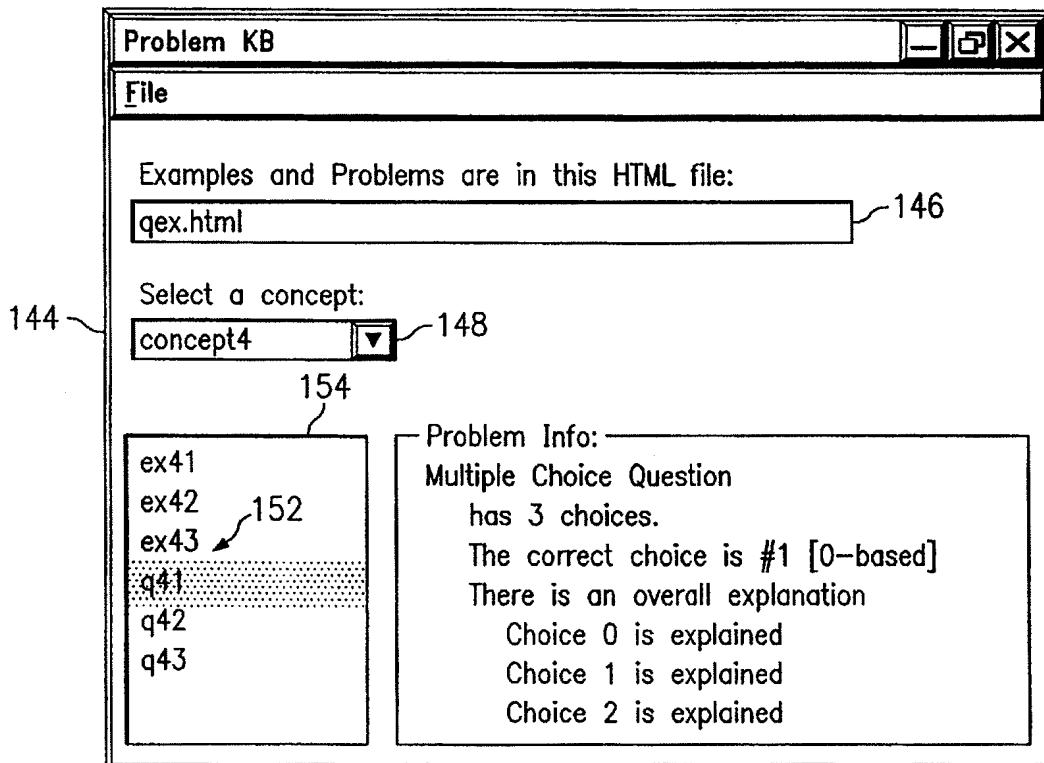


FIG. 19

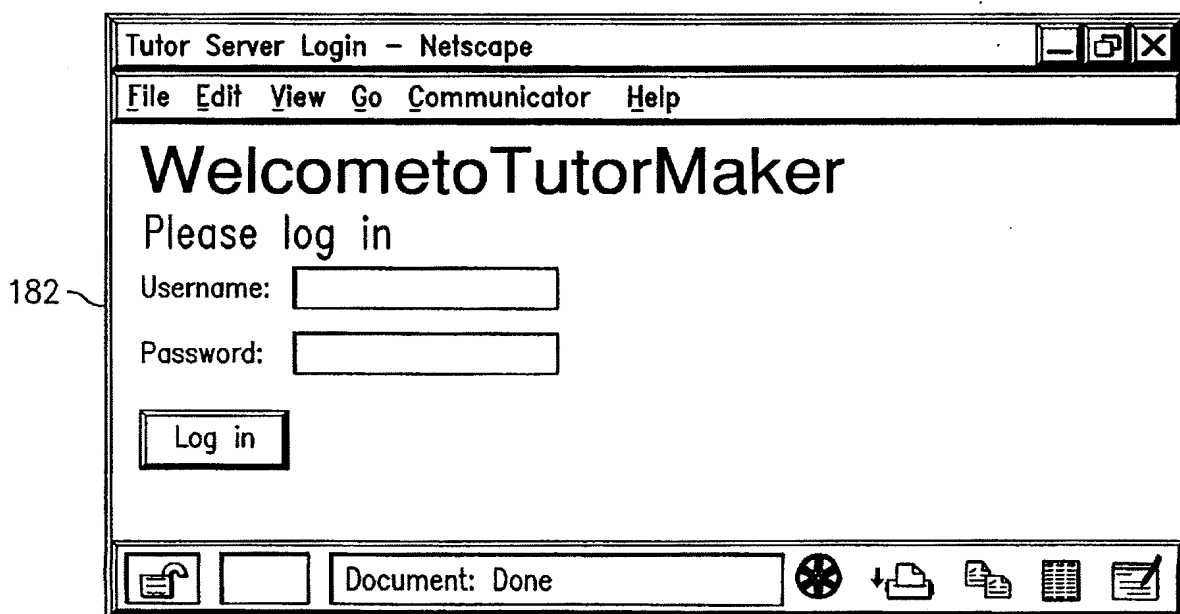


FIG. 22

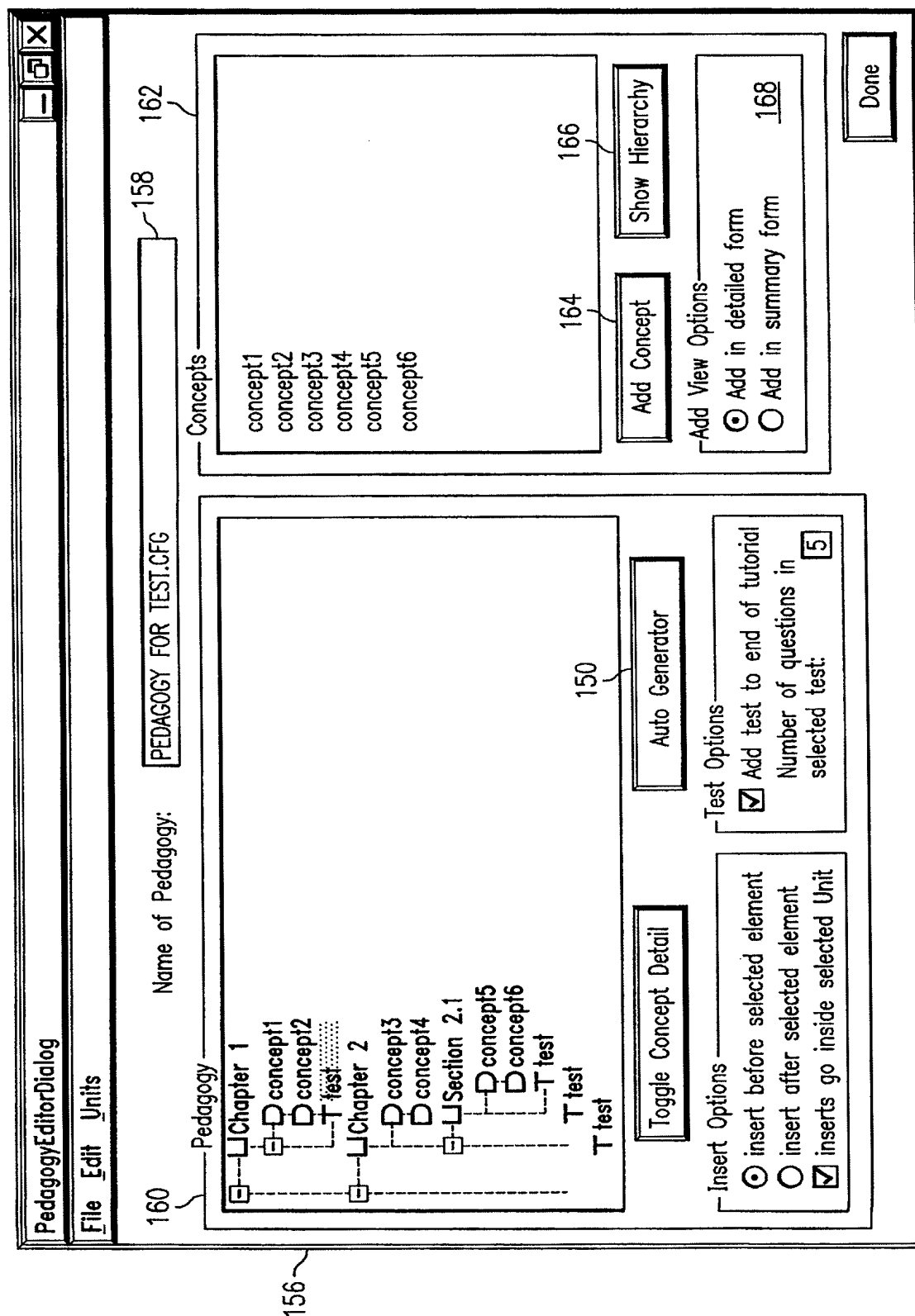


FIG. 20

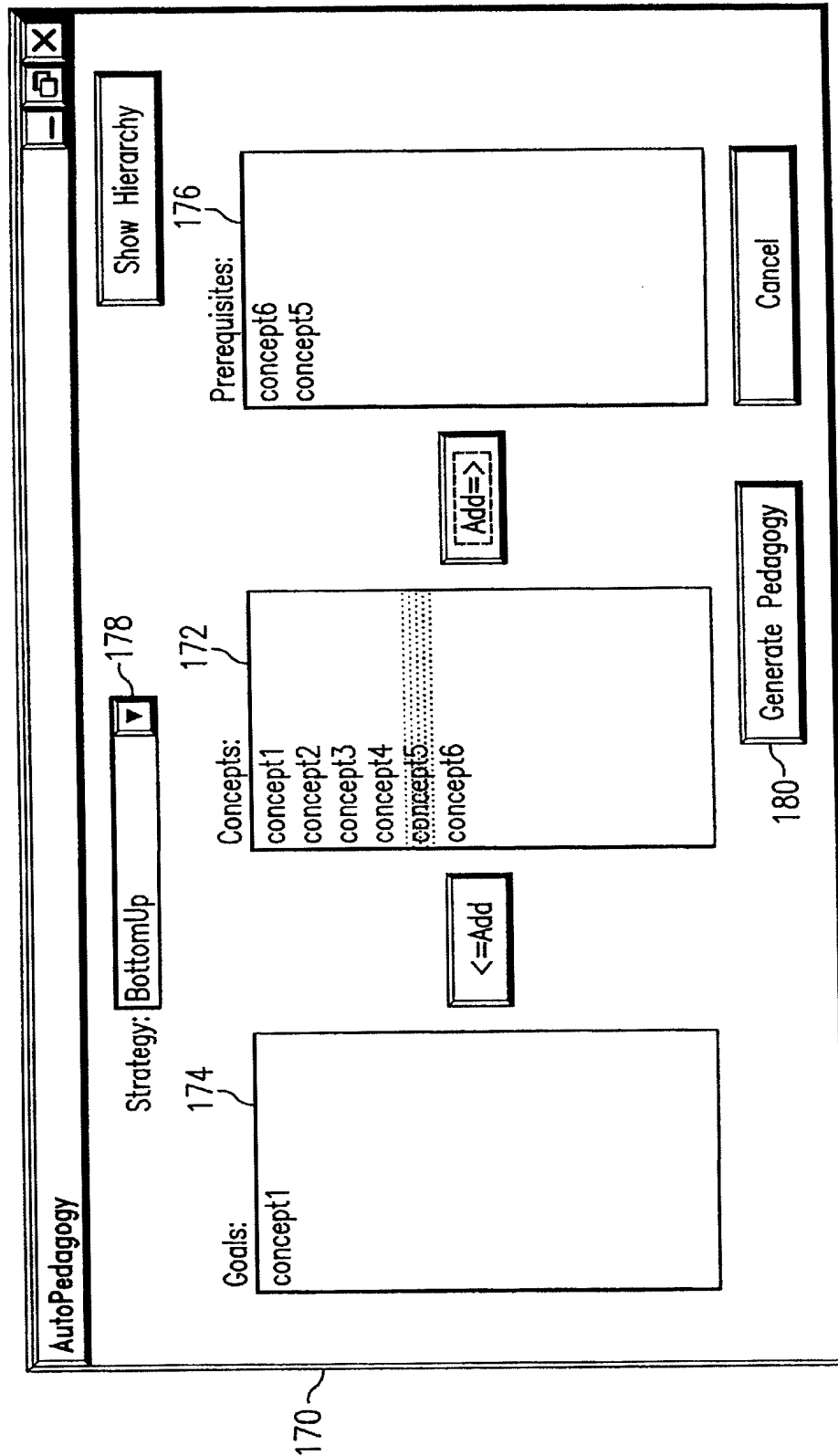


FIG. 21

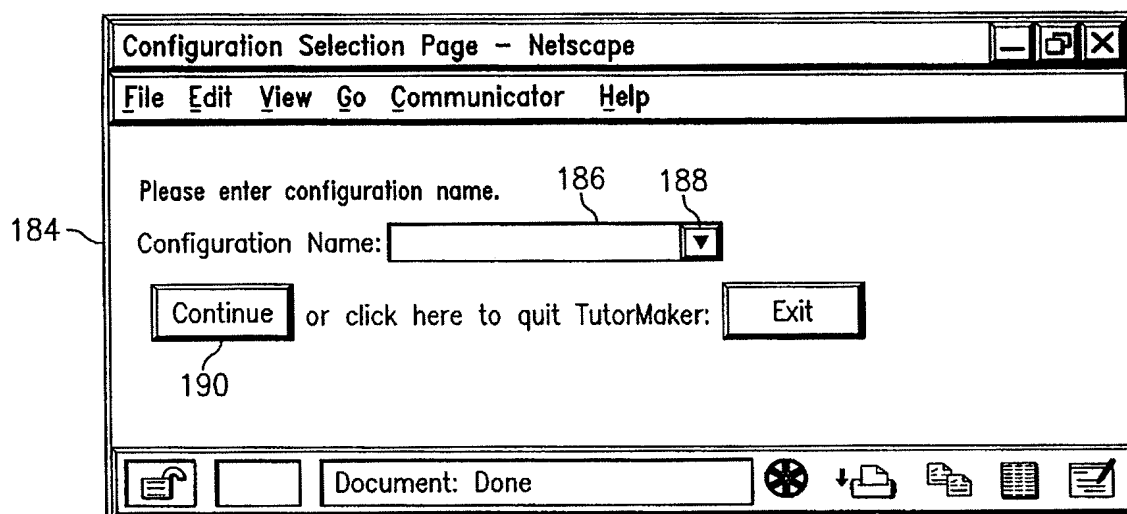


FIG. 23

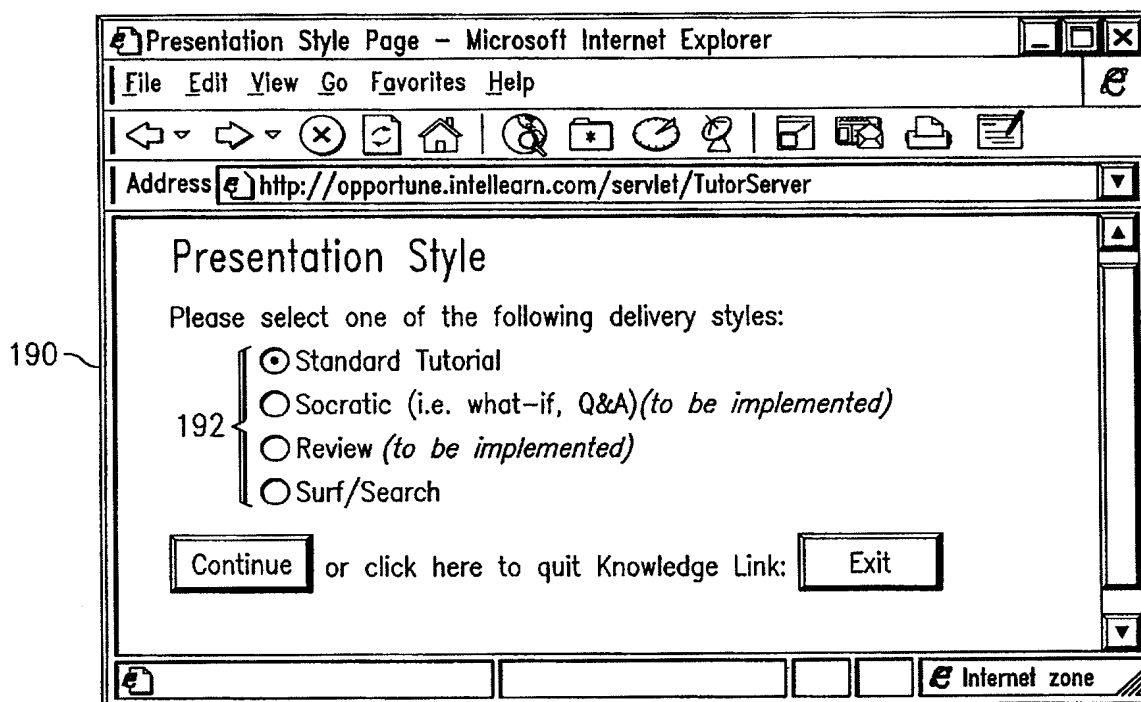


FIG. 24

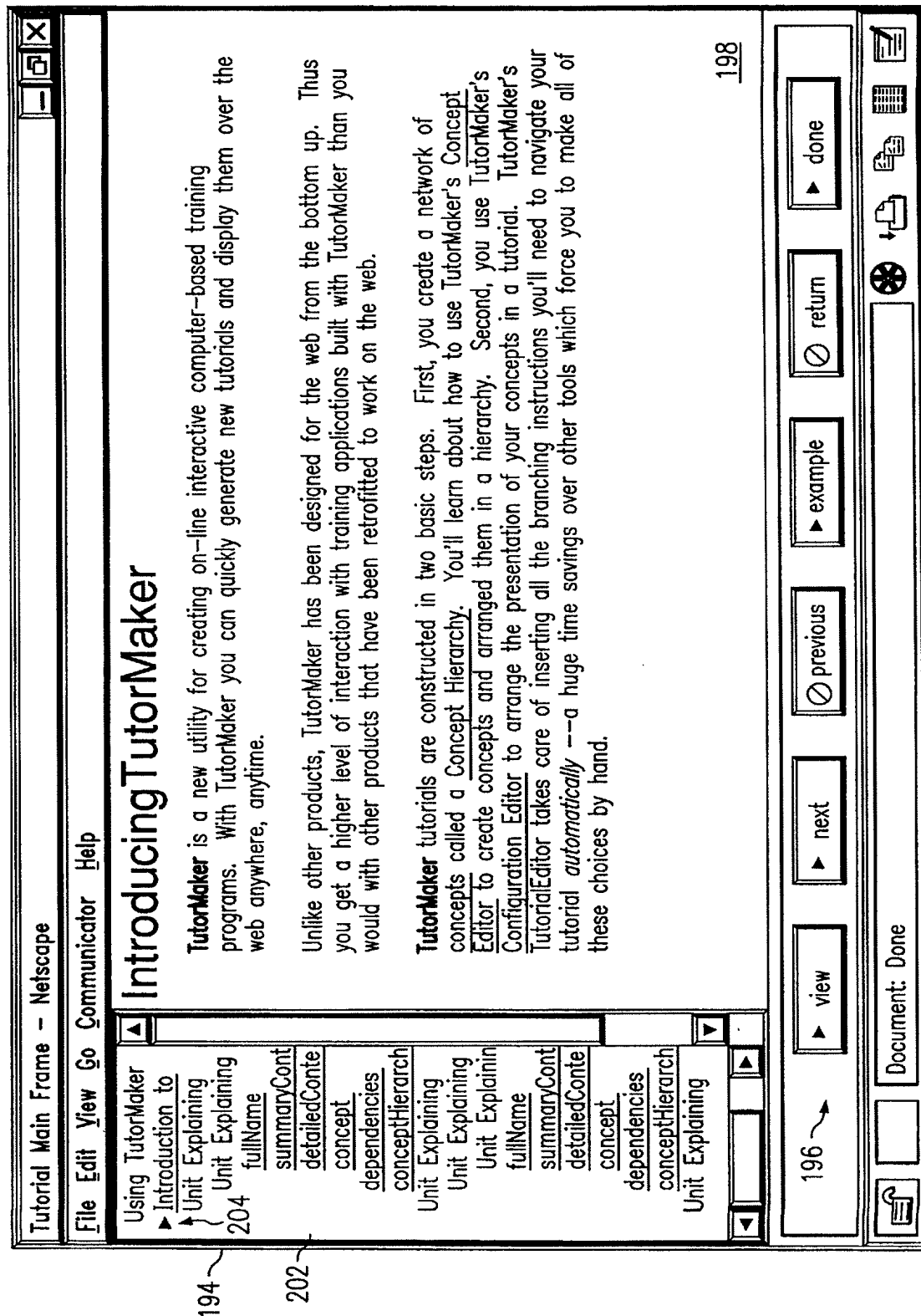
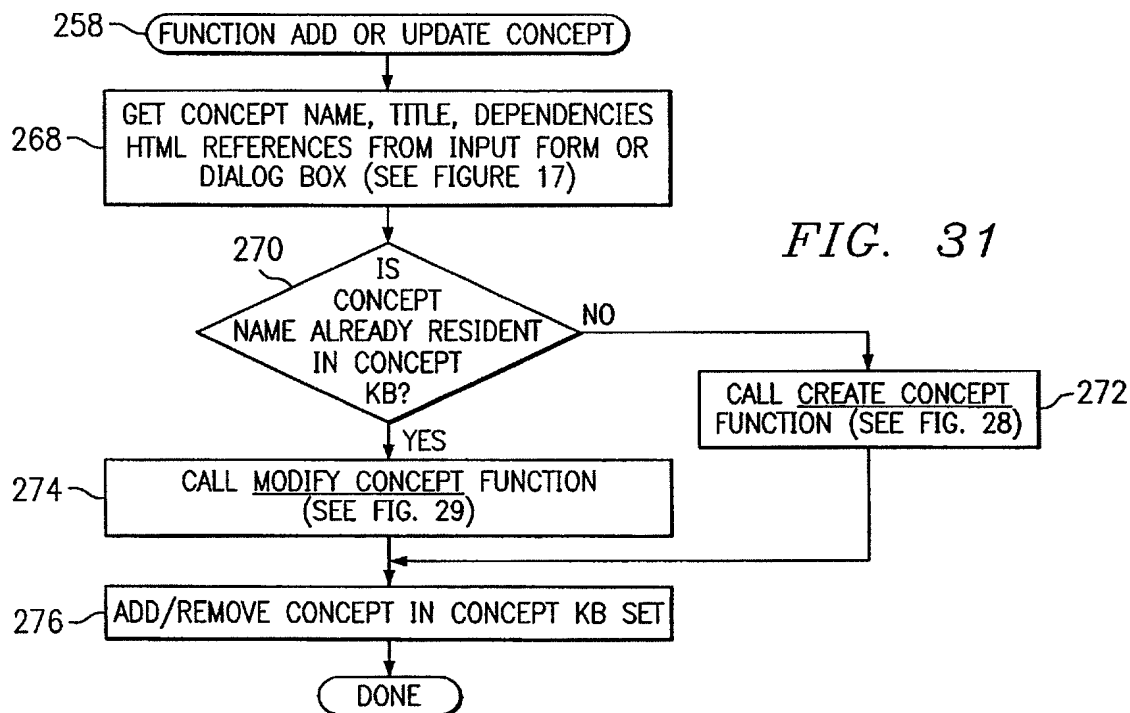
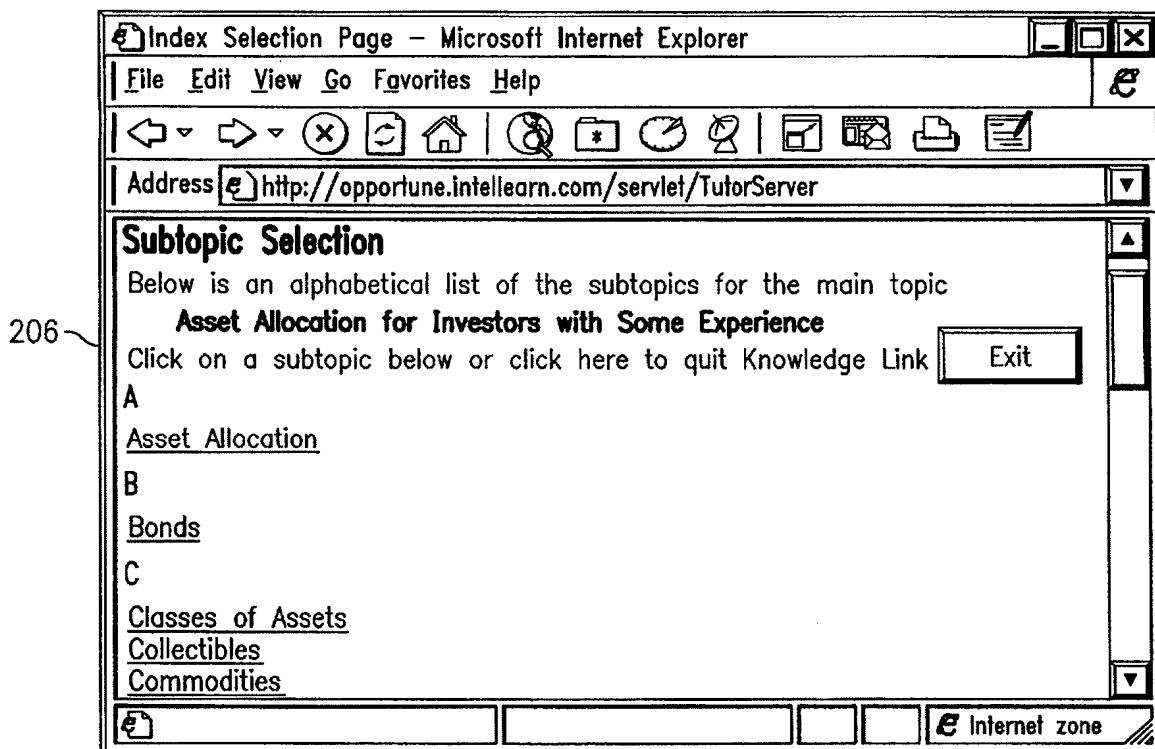


FIG. 25

FIG. 26



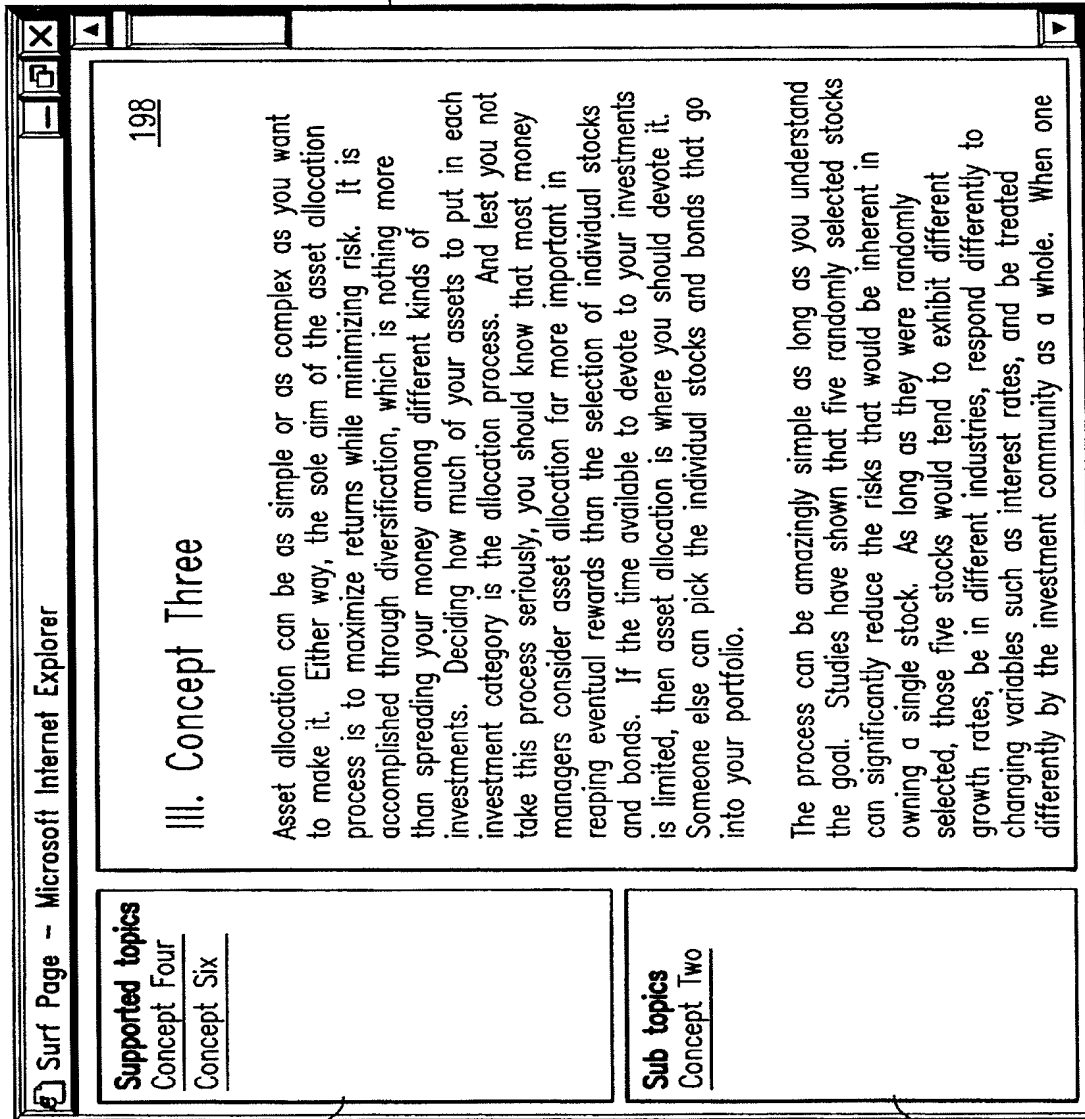
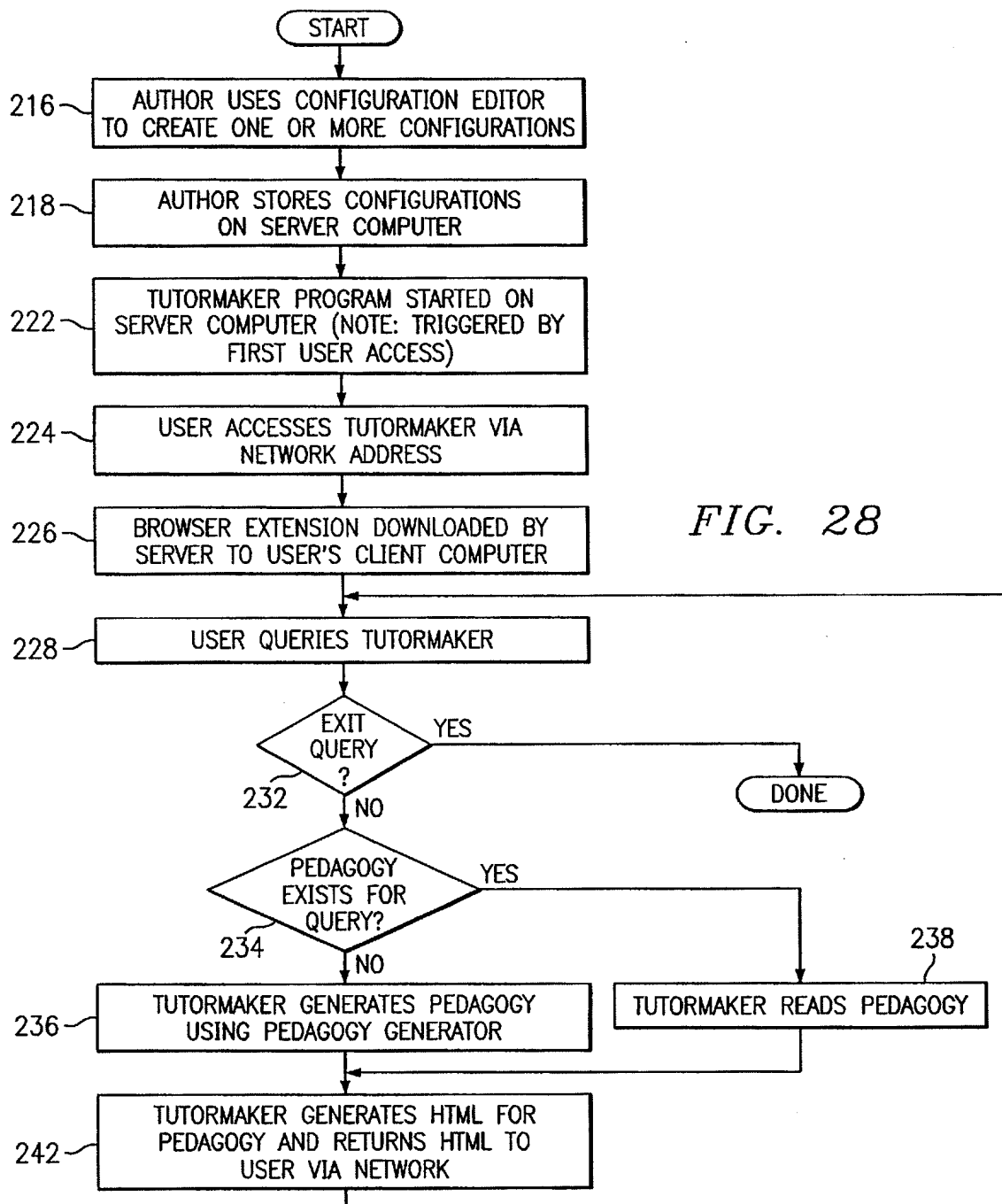
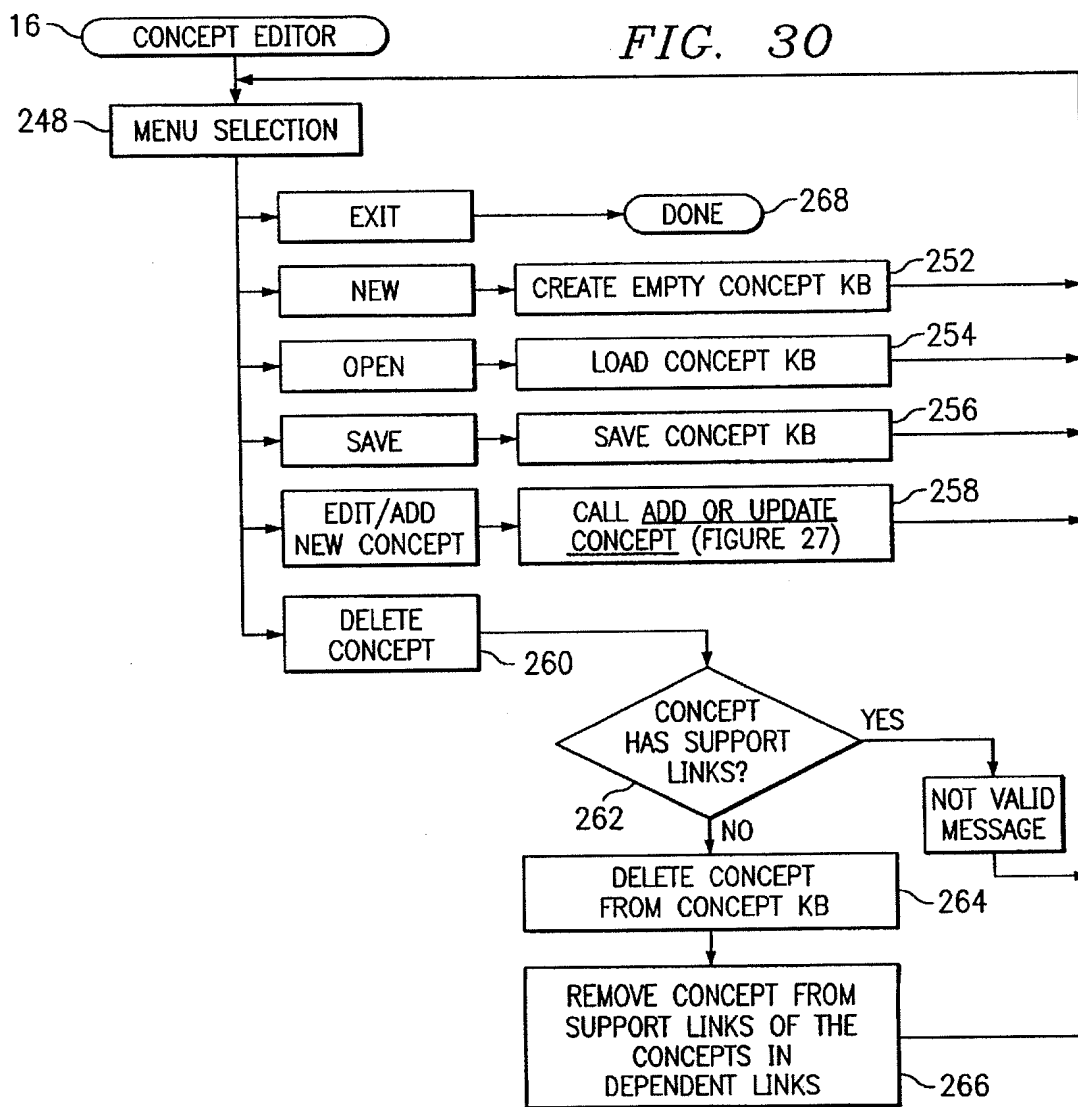
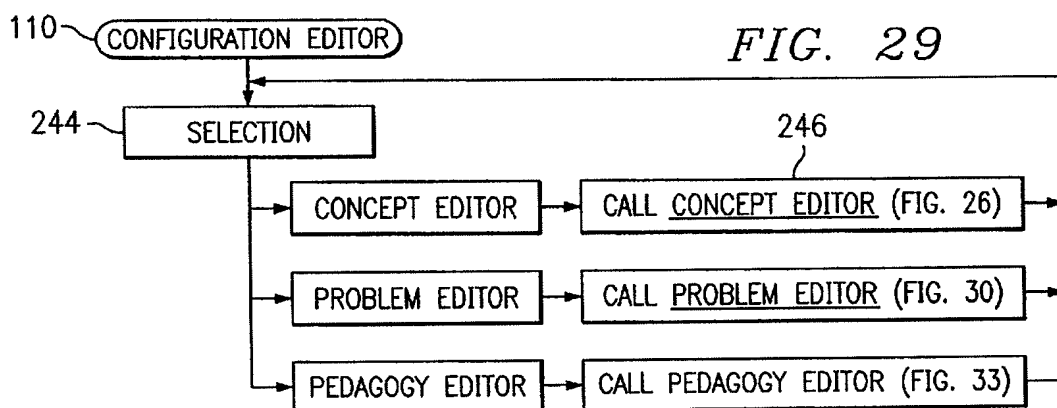


FIG. 27





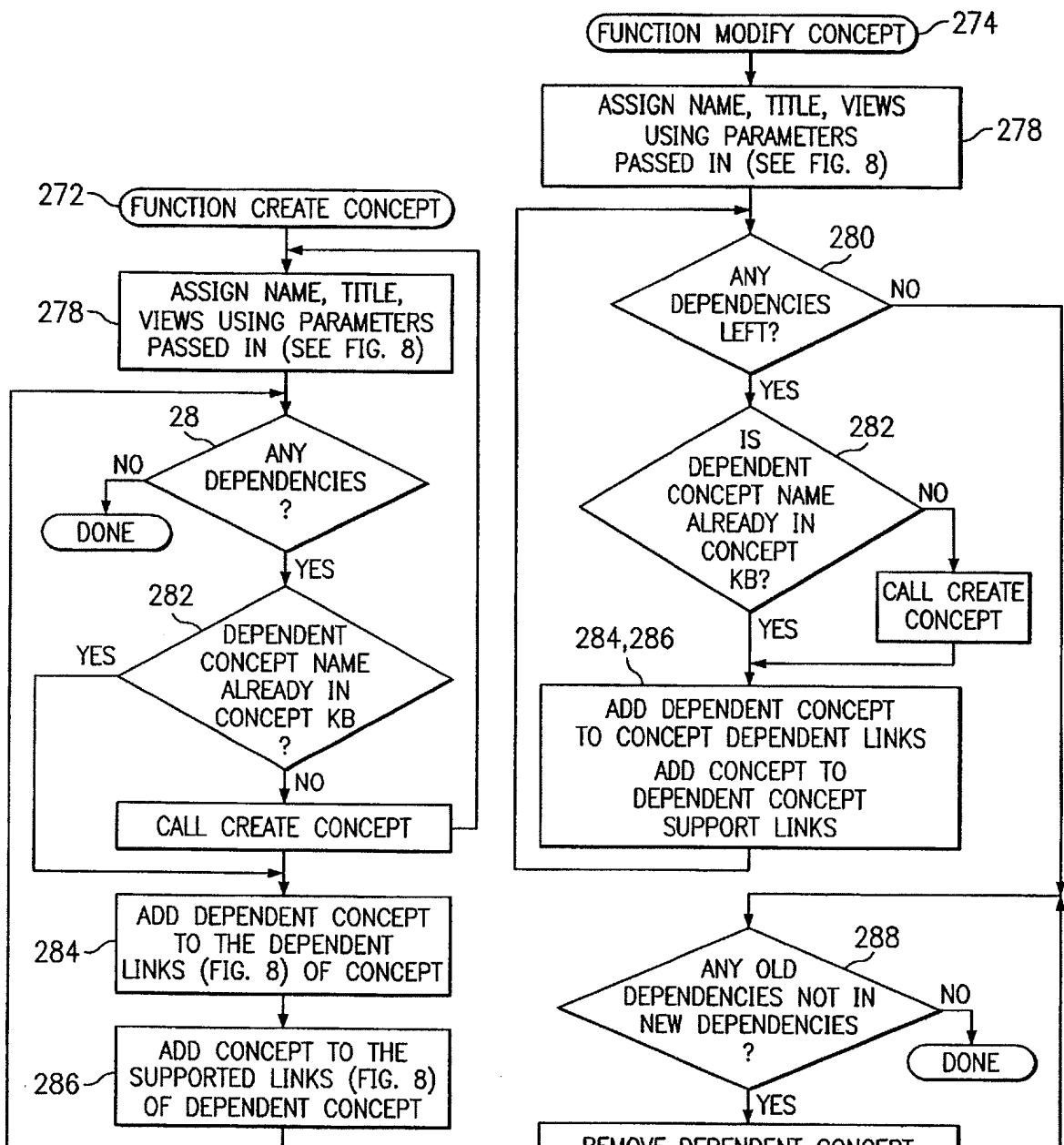


FIG. 32

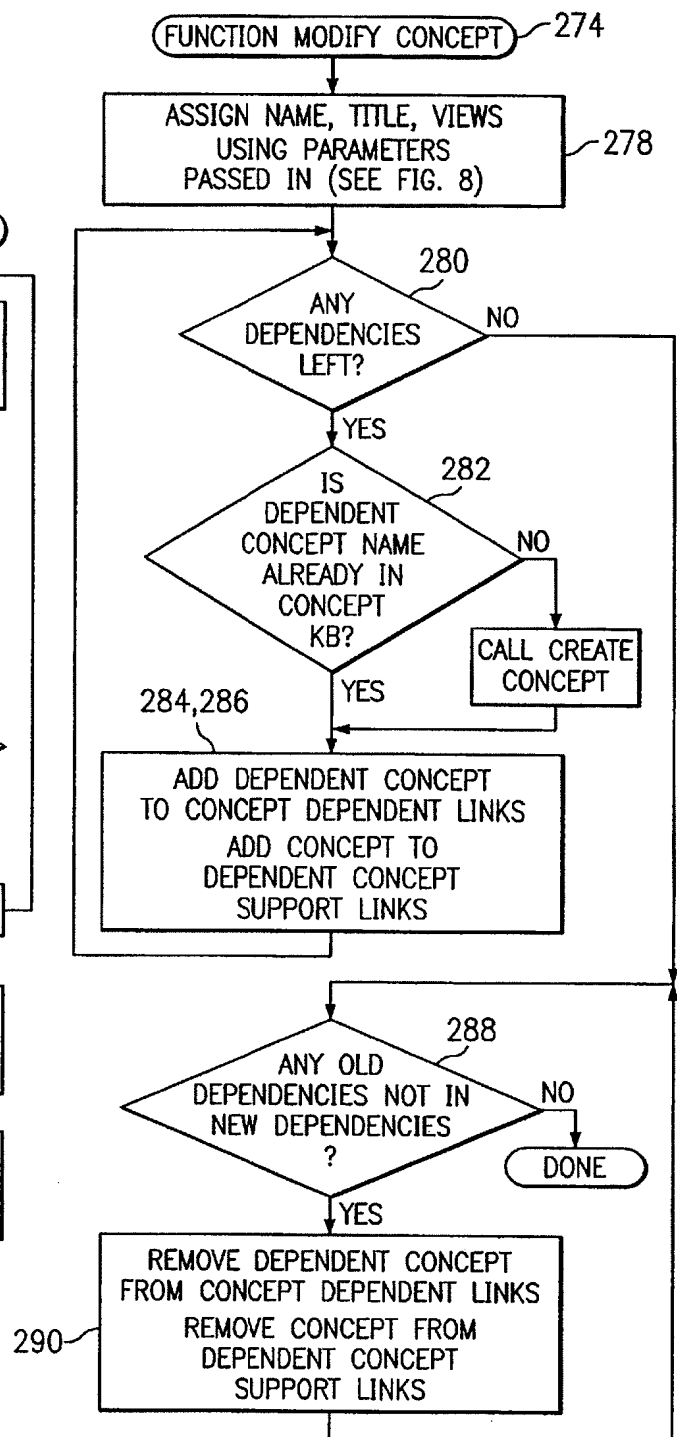


FIG. 33

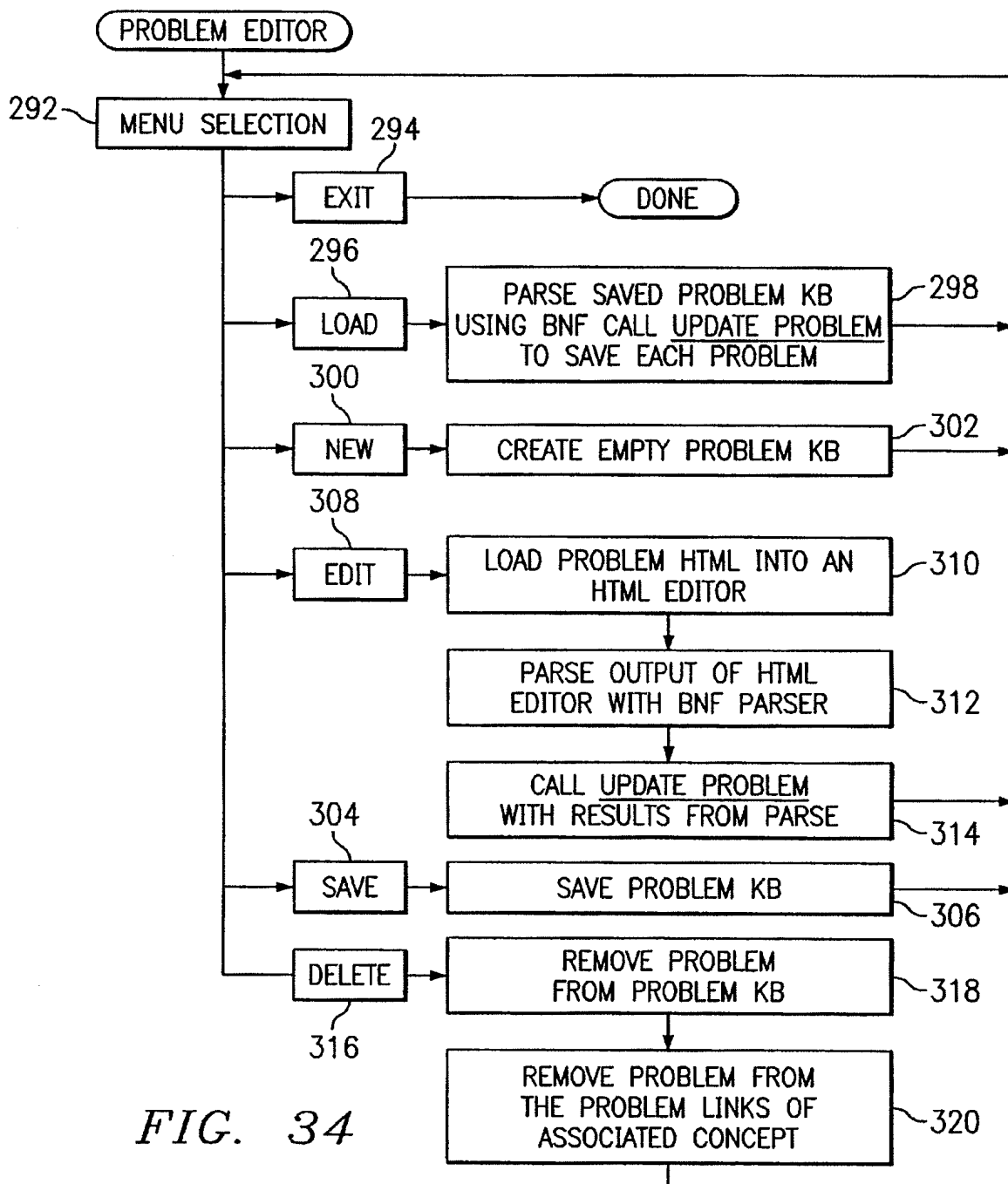


FIG. 34

FIG. 35

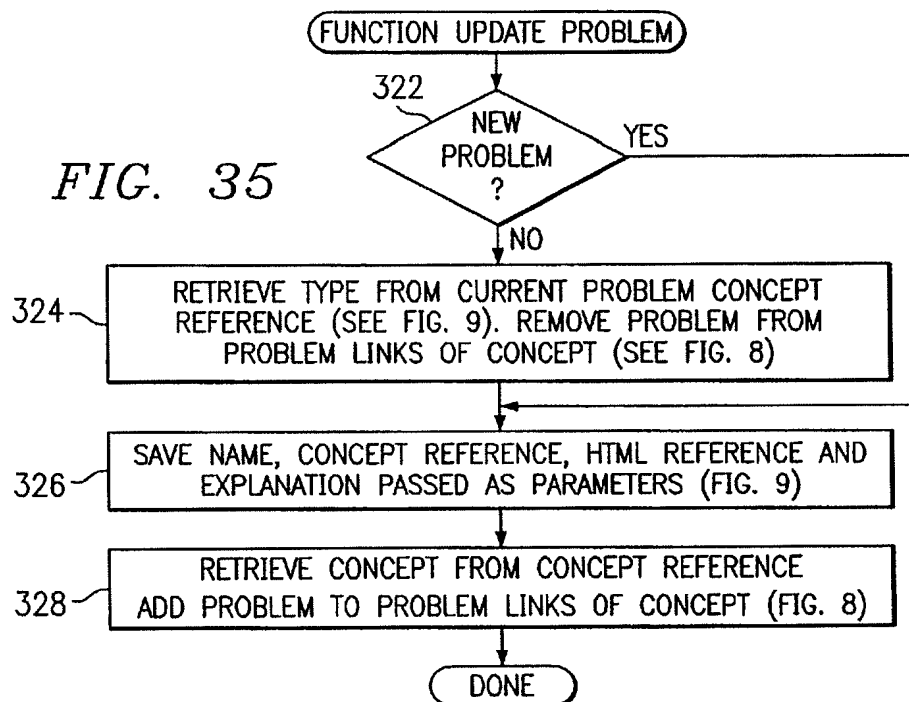
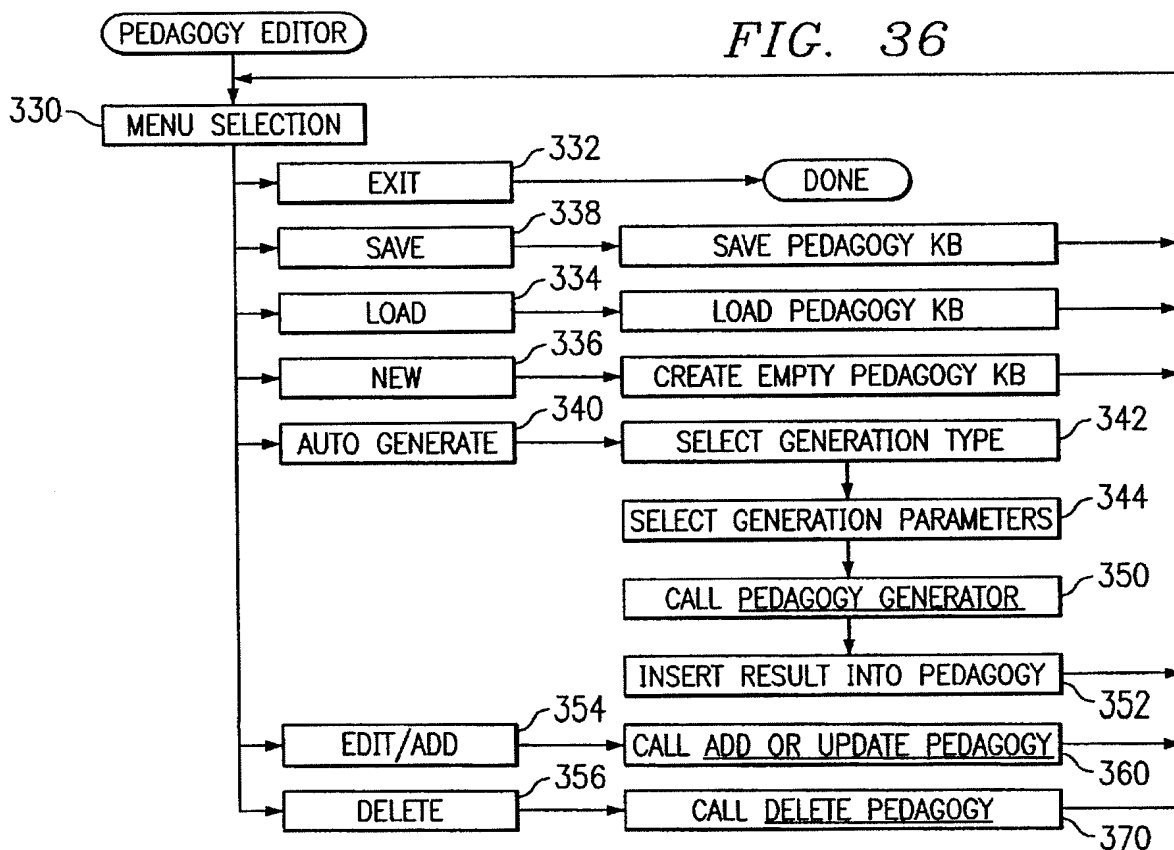


FIG. 36



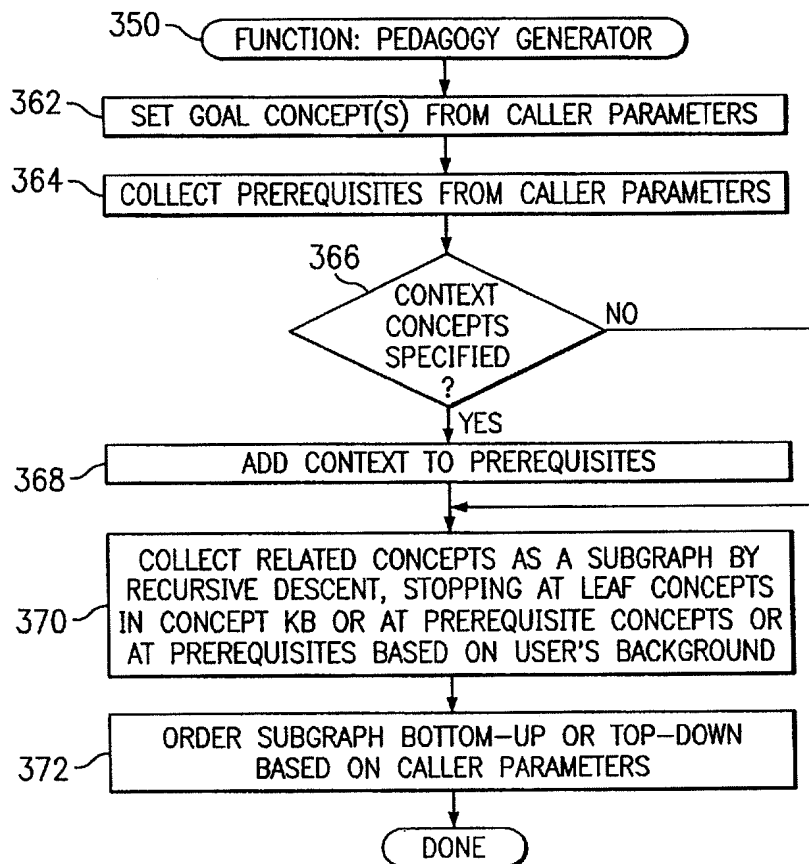


FIG. 37A

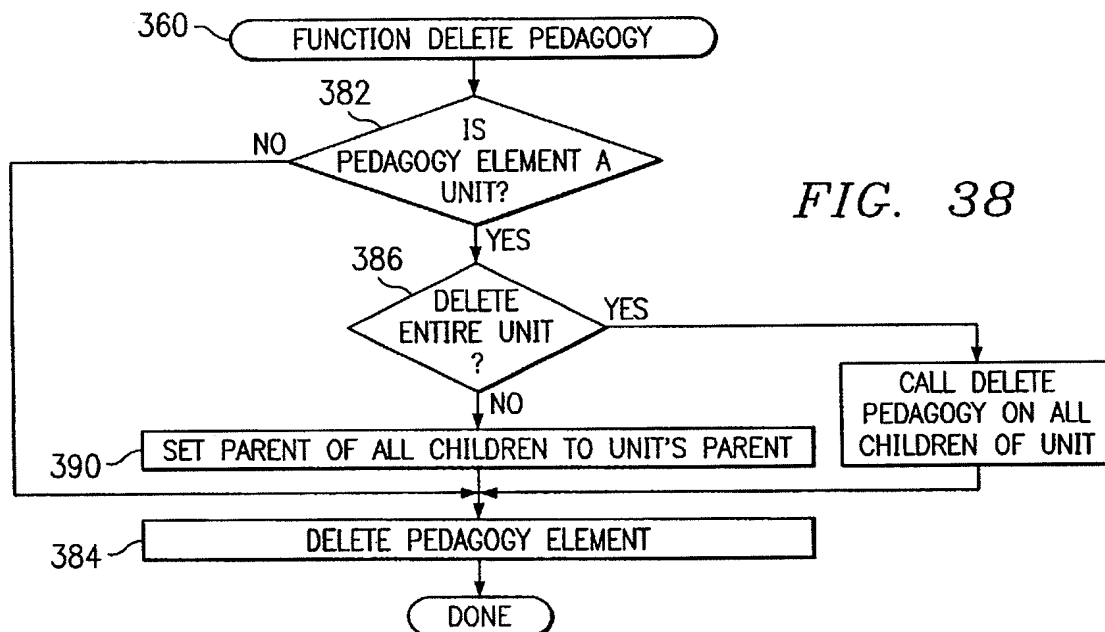


FIG. 38

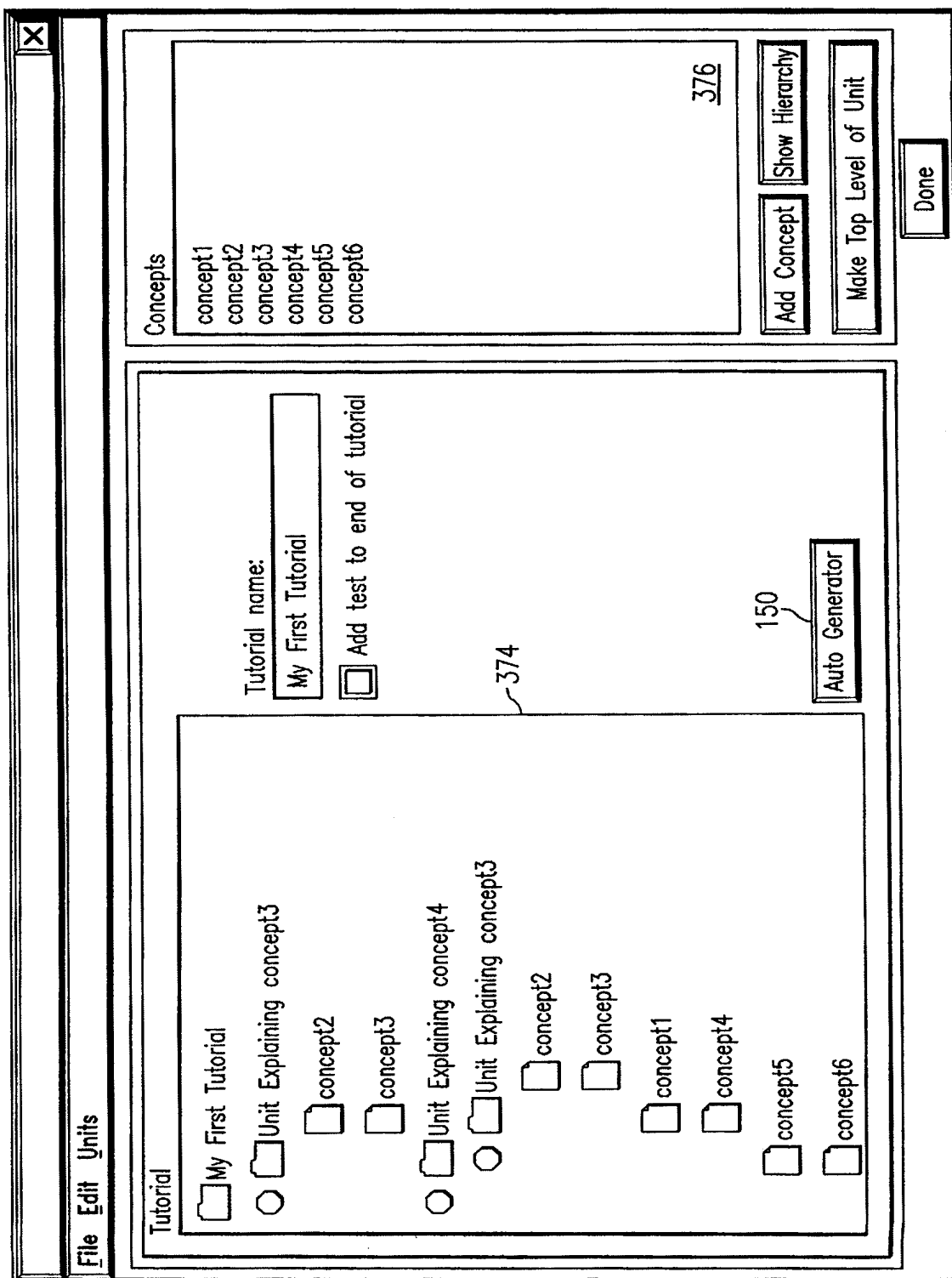


FIG. 37B

FIG. 37C

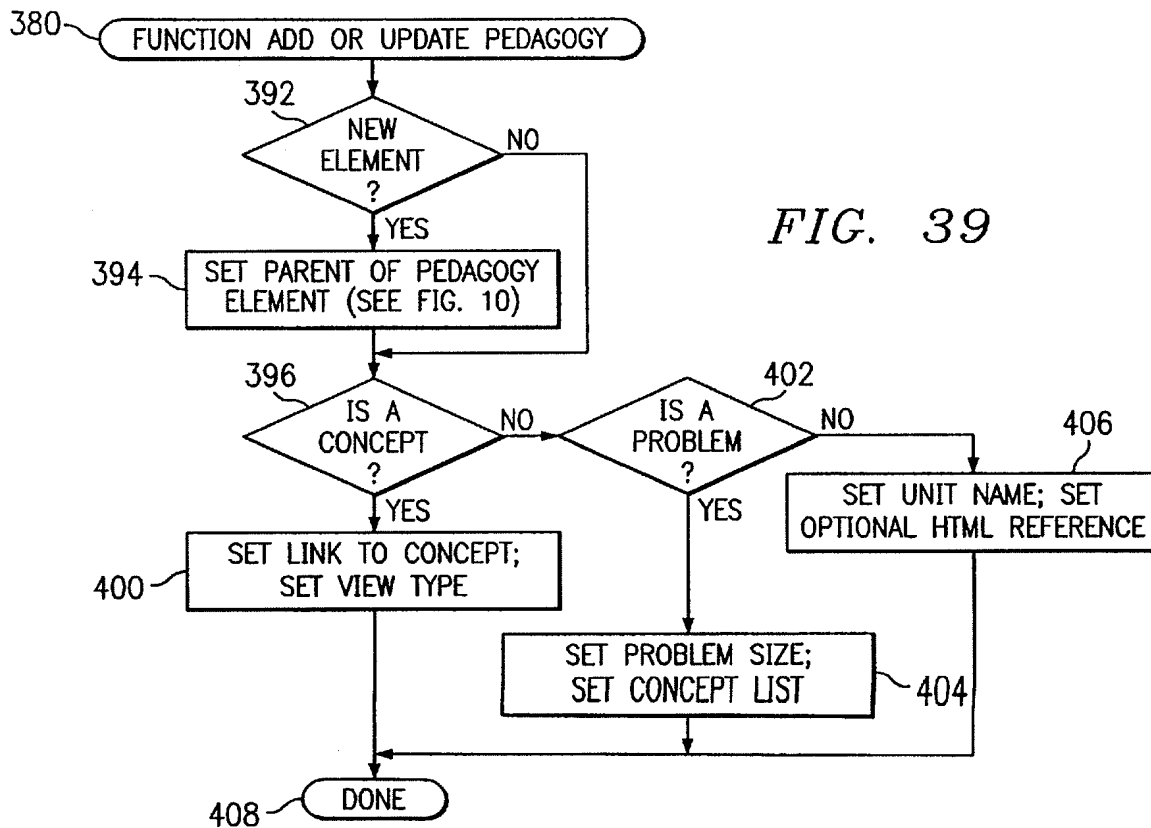
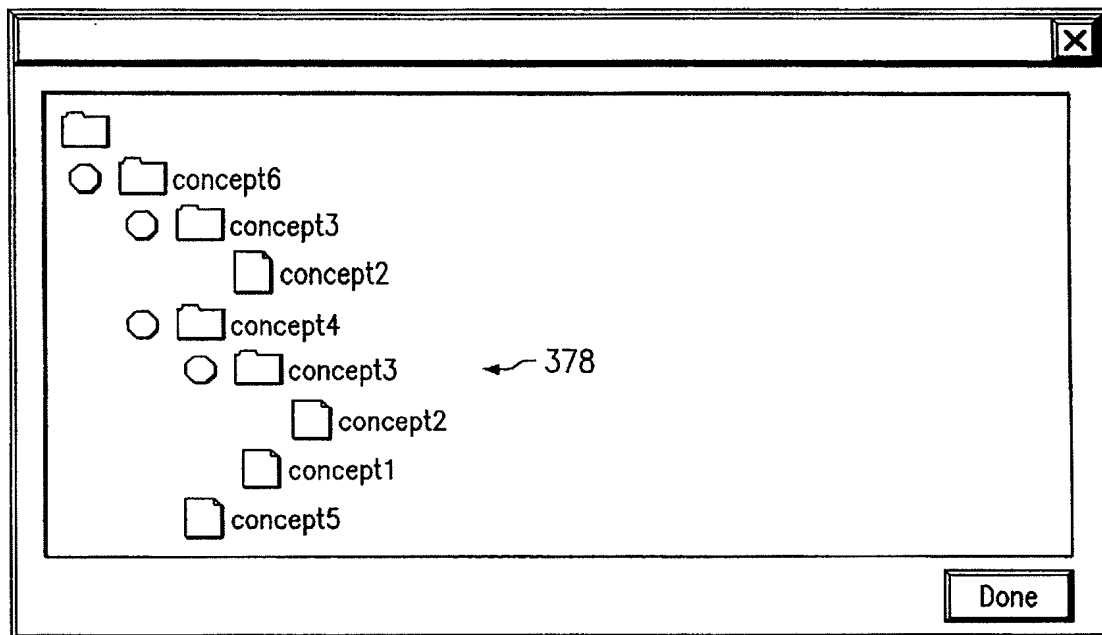


FIG. 40

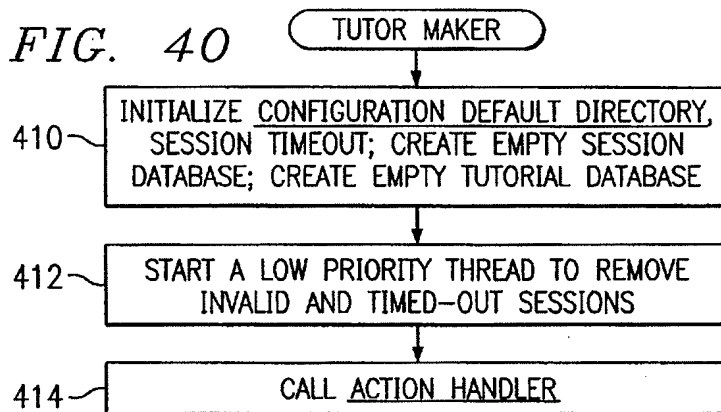
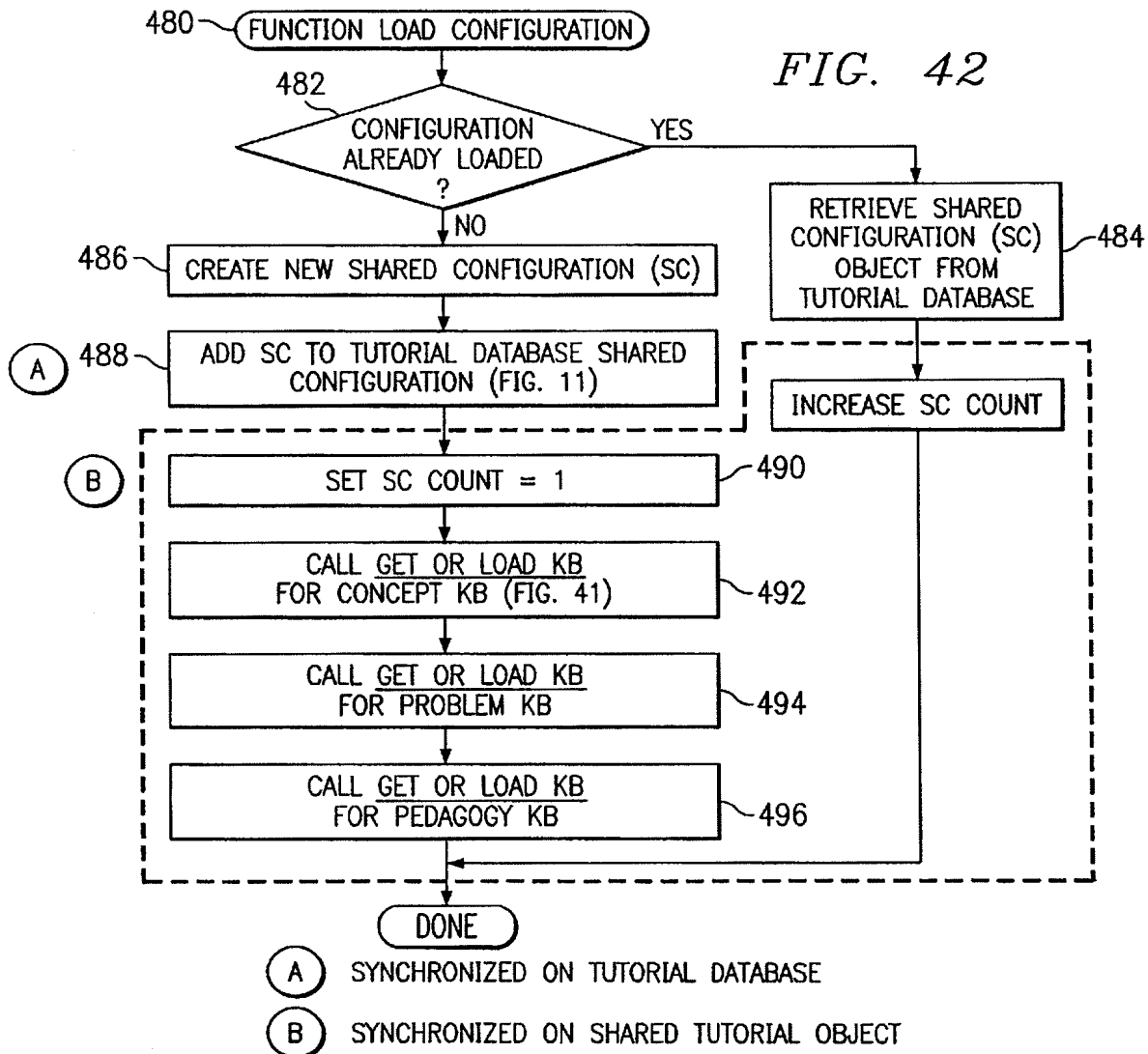
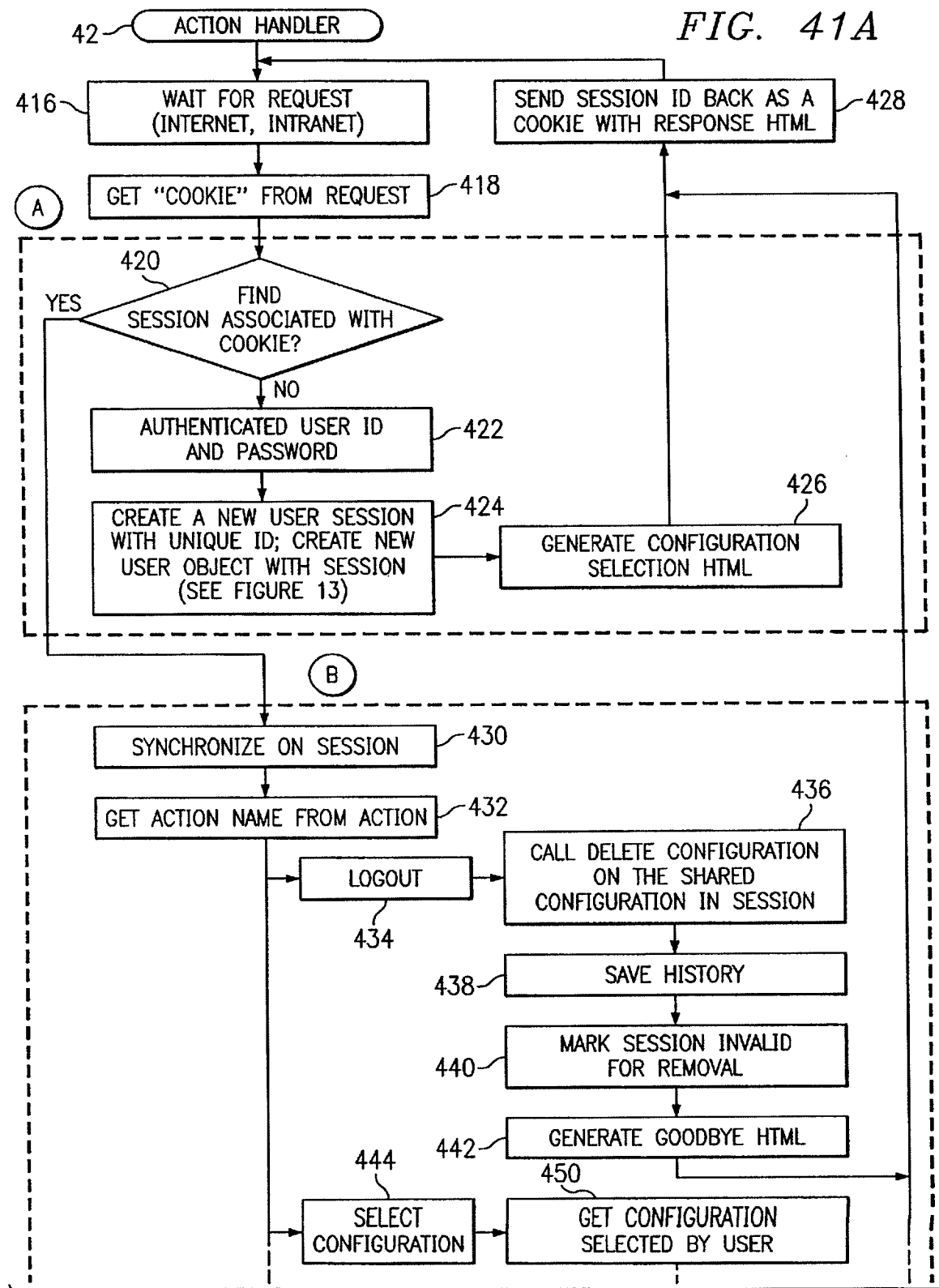


FIG. 42

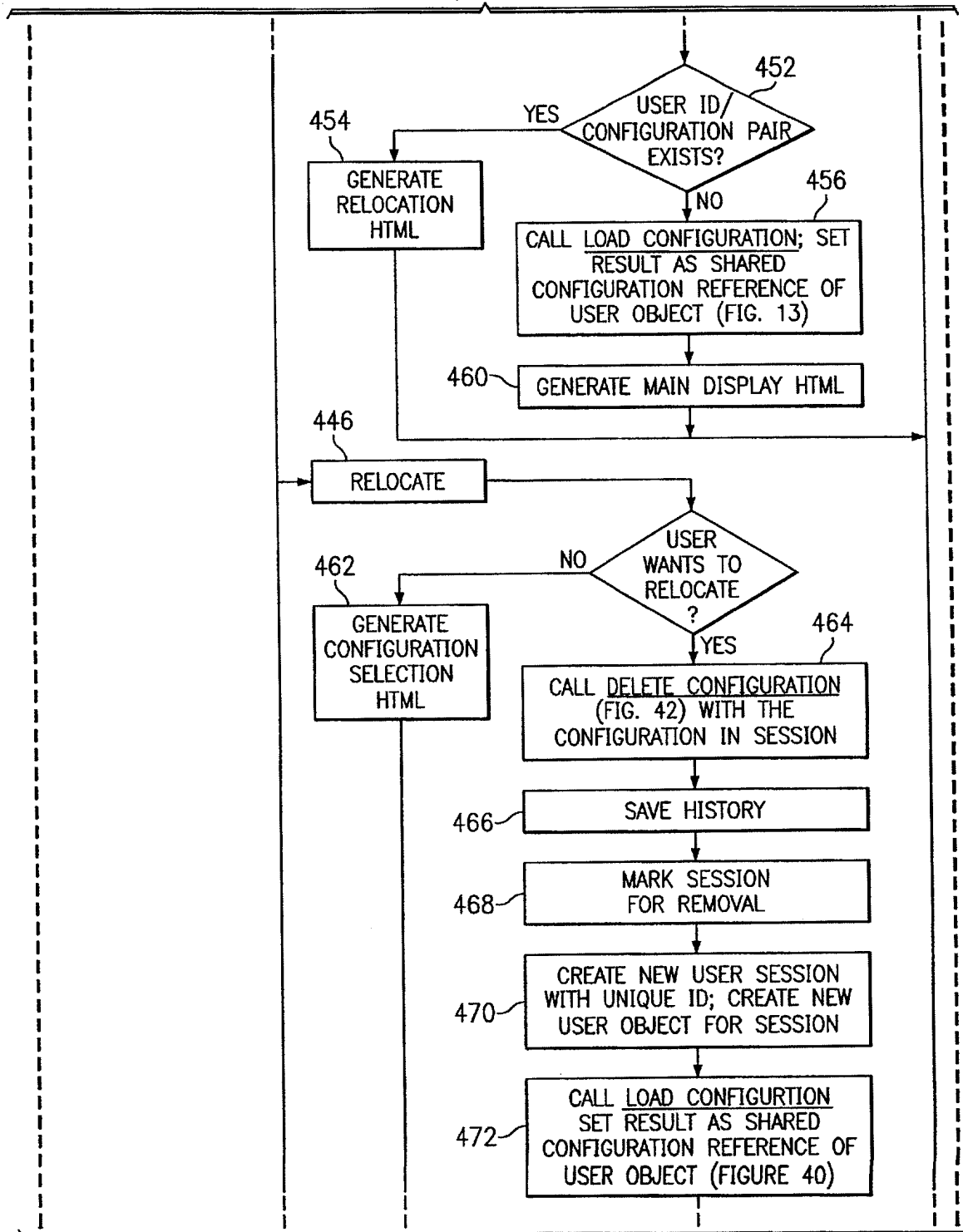




TO FIG. 41B

FIG. 41B

FROM FIG. 41A



TO FIG. 41C

FIG. 41C

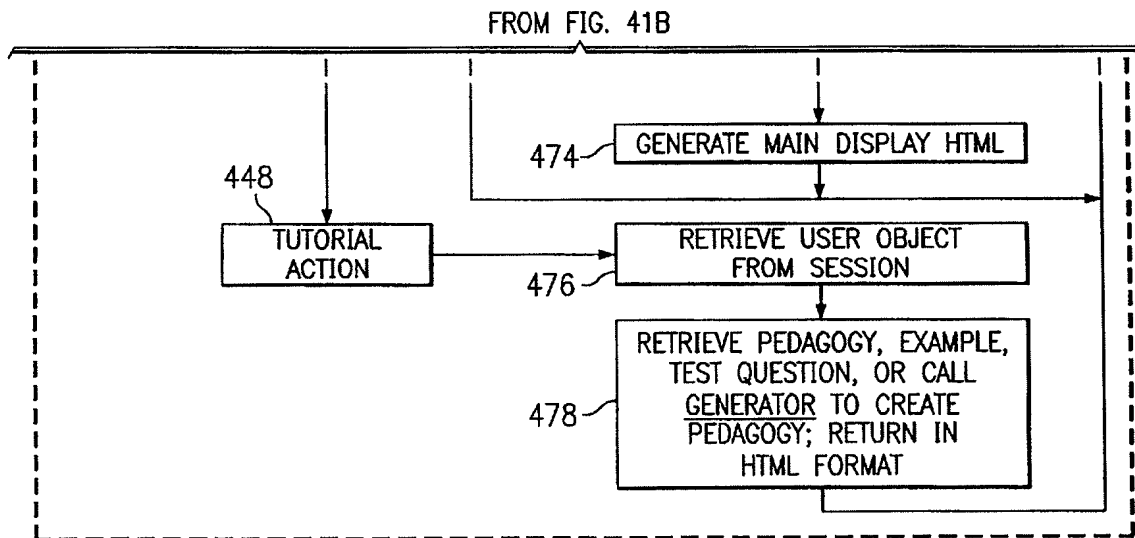


FIG. 44

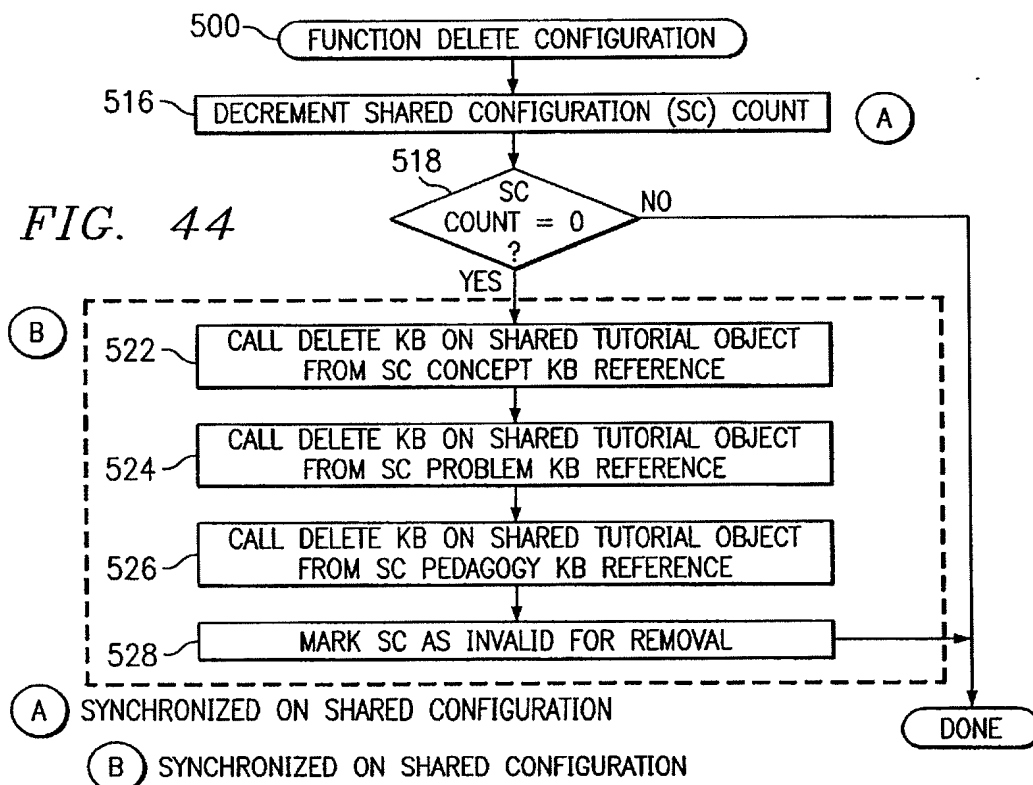


FIG. 43

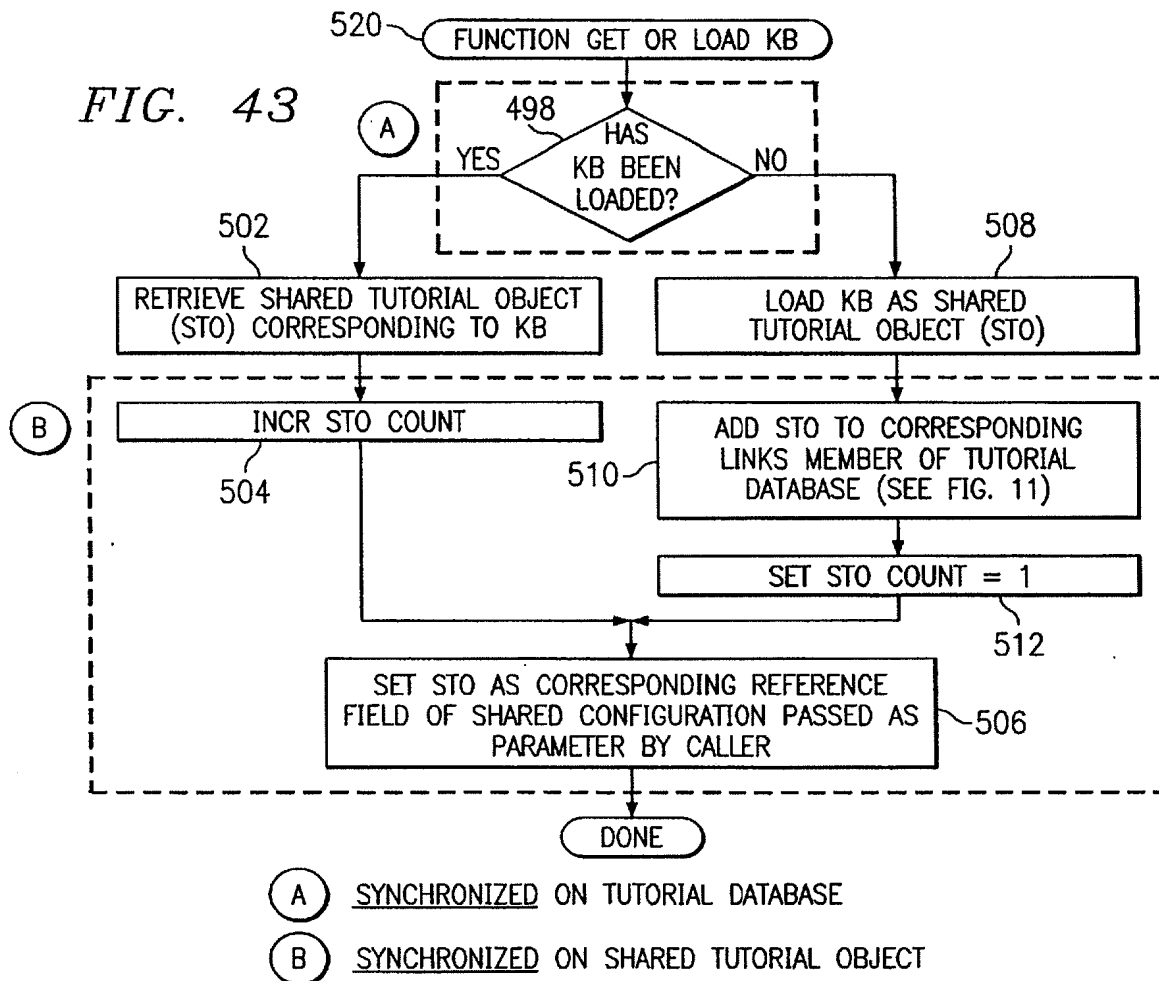
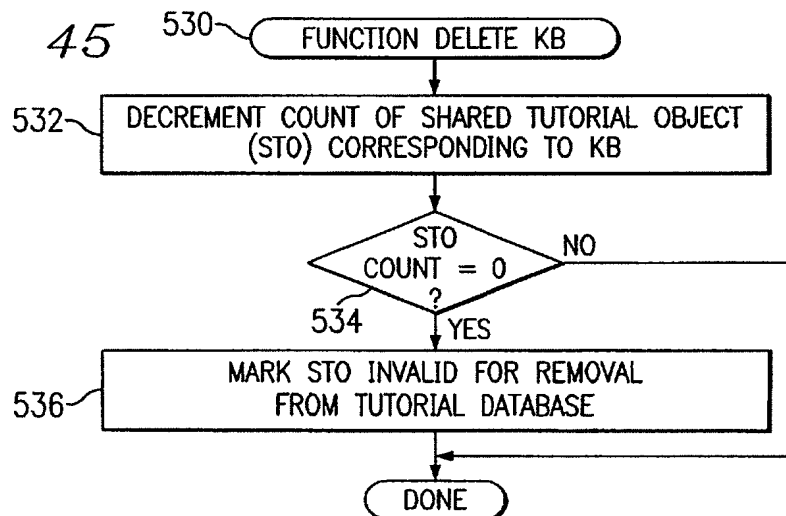


FIG. 45



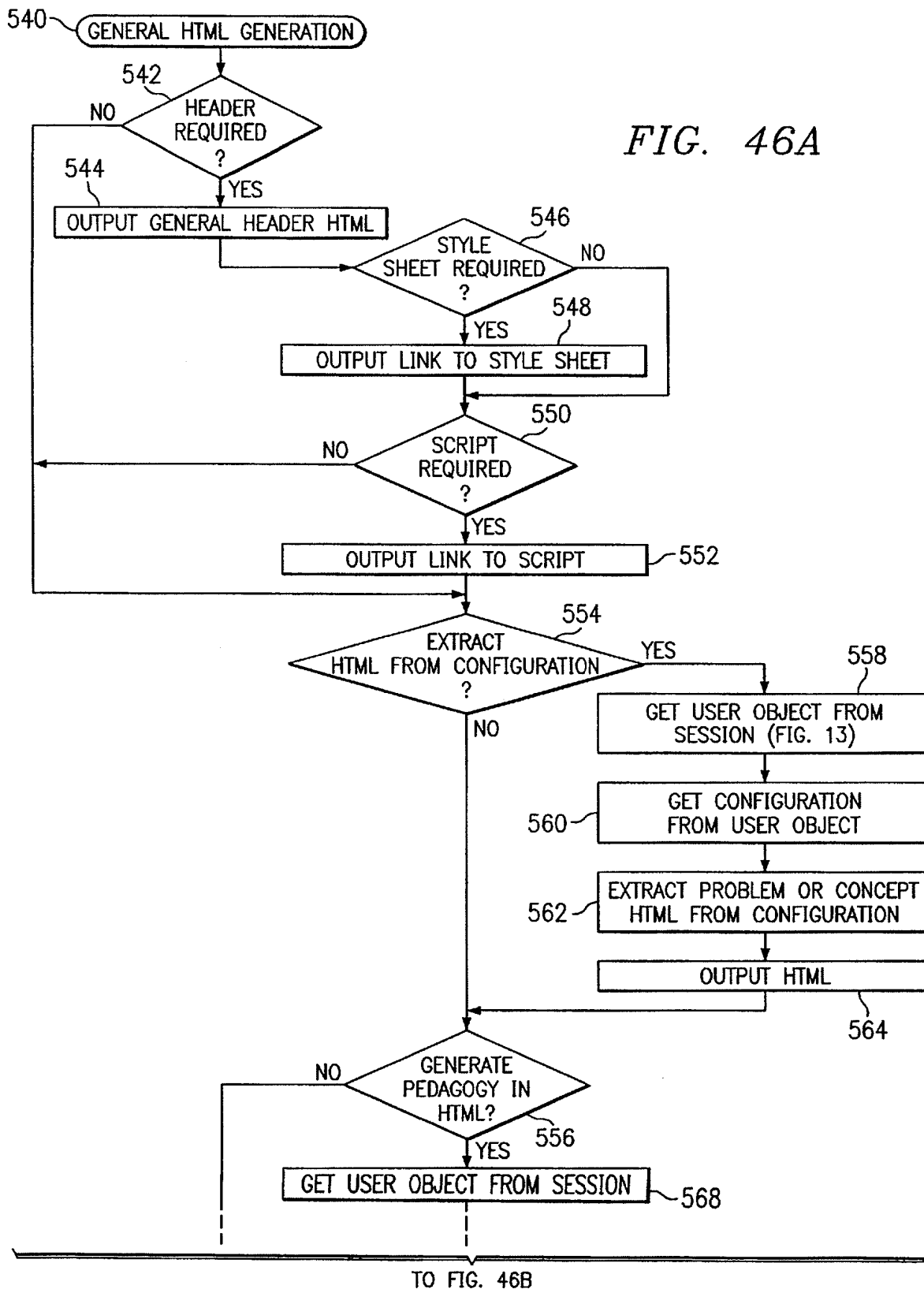


FIG. 46B

FROM FIG. 46A

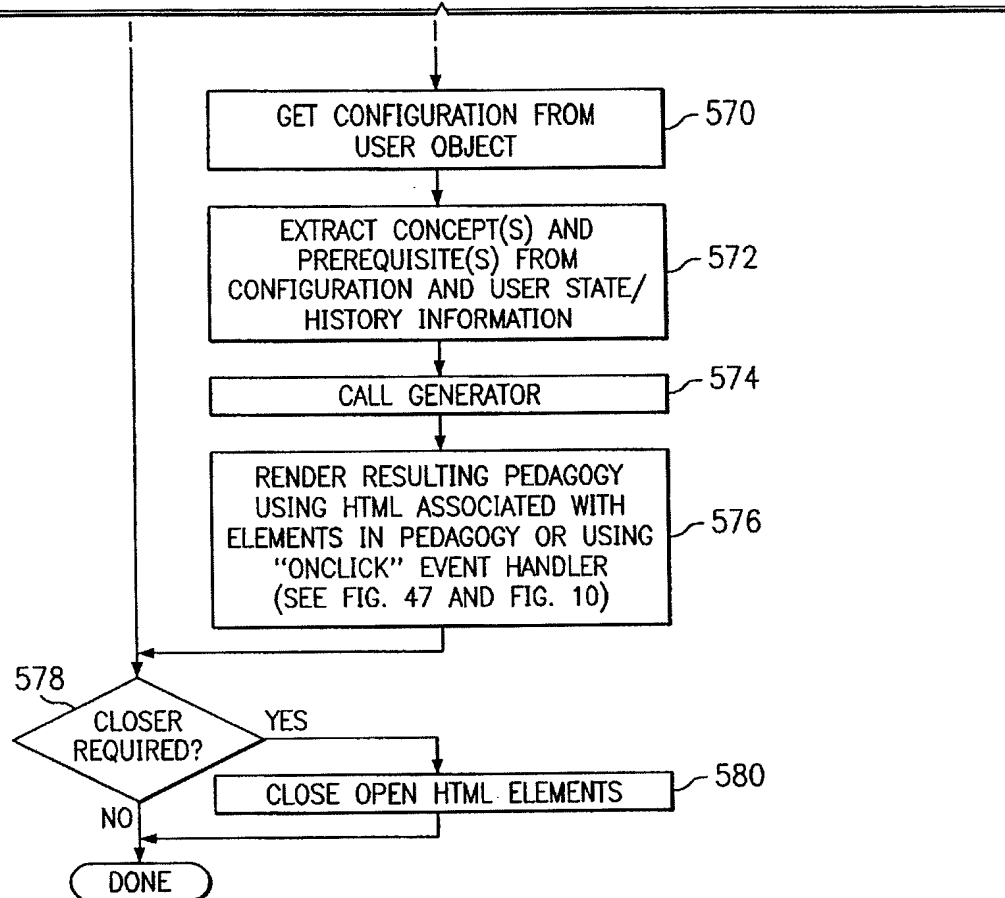
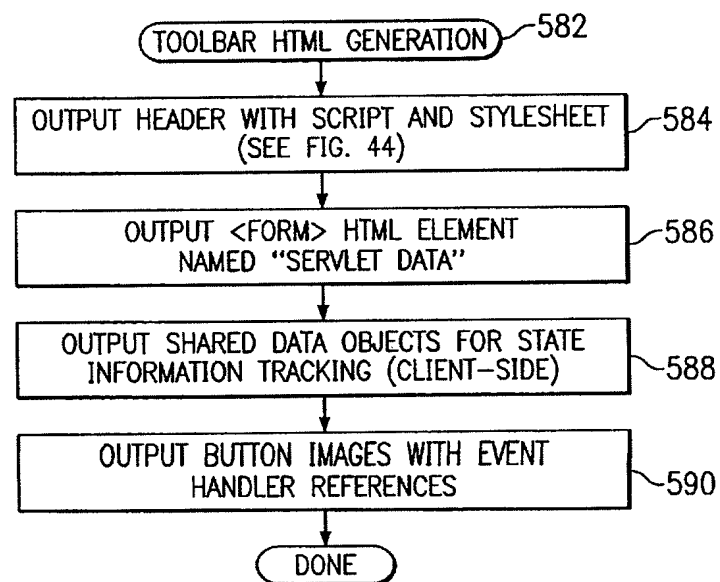
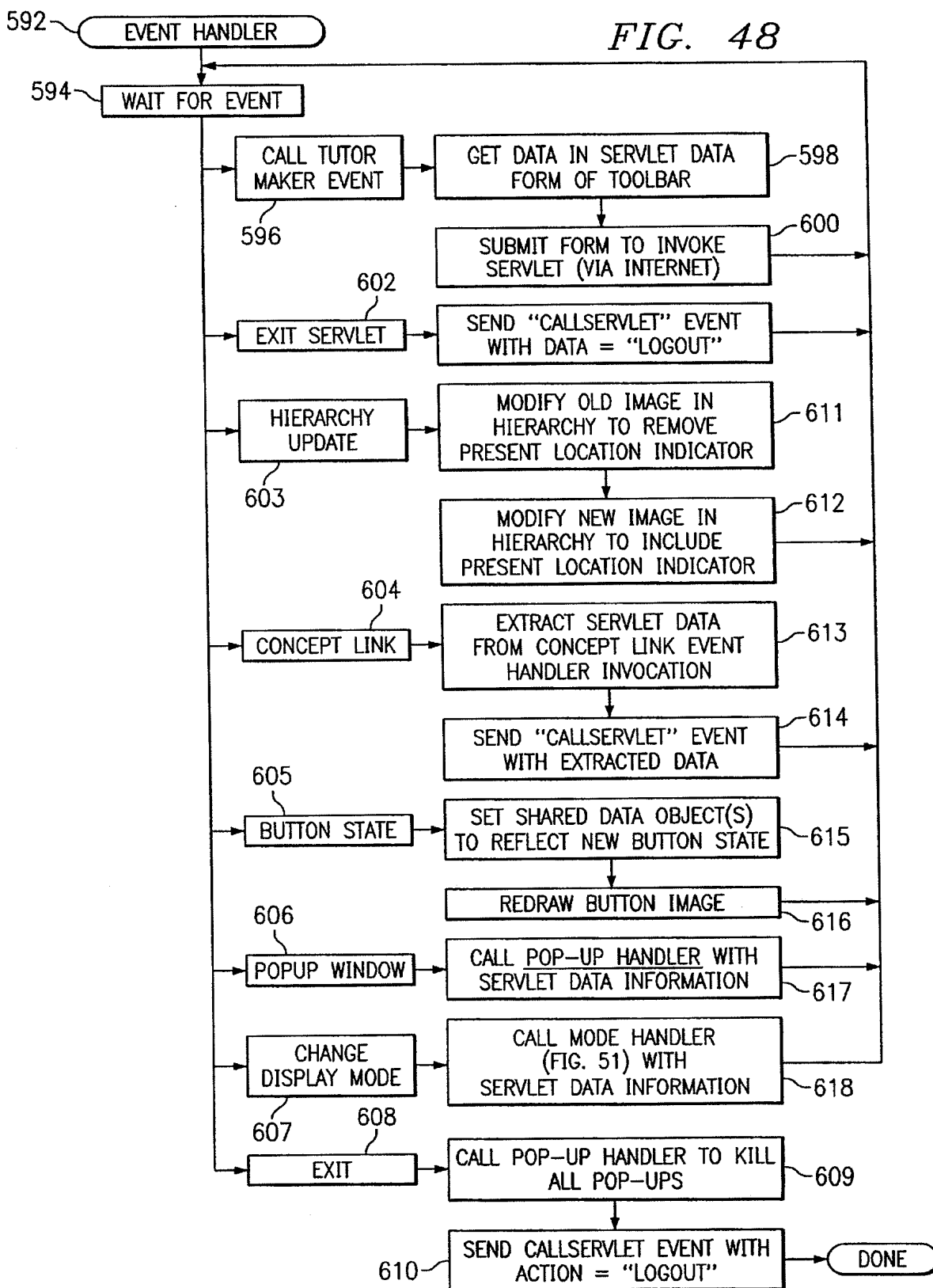


FIG. 47





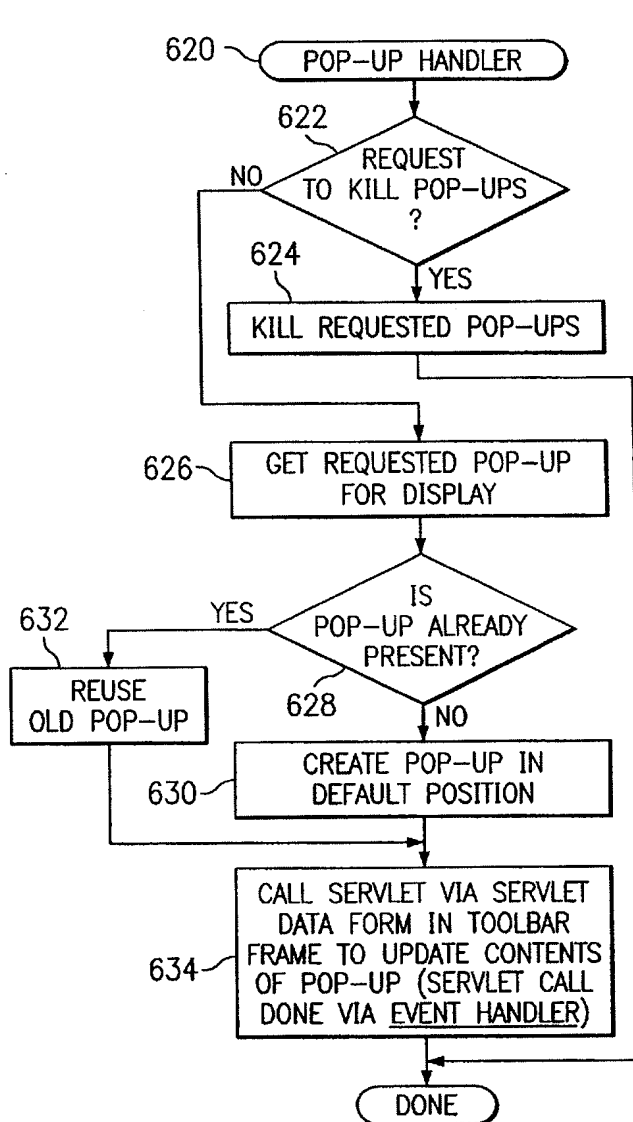


FIG. 49

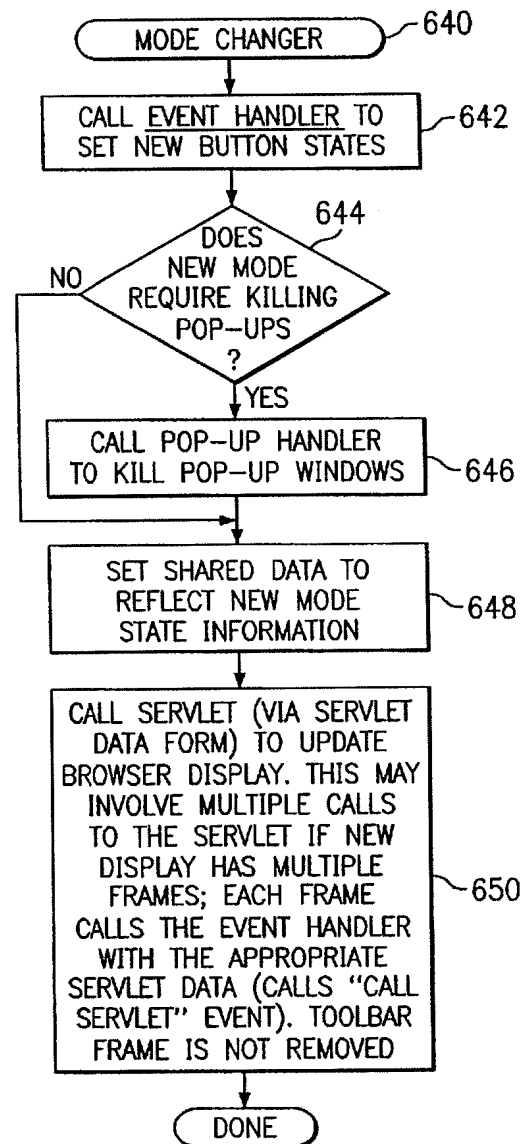


FIG. 50

1

SYSTEM AND METHOD FOR DYNAMIC KNOWLEDGE GENERATION AND DISTRIBUTION

GOVERNMENT LICENSE RIGHTS

The United States government has a paid-up license in this invention as provided for in FAAR 52.227-11 subsections B and C.

TECHNICAL FIELD OF THE INVENTION

The present invention relates generally to knowledge creation and distribution systems and methods, and more particularly, a system and method using software programming to create a knowledge base, organize the knowledge base or some subset thereof, and distribute the knowledge in various formats, including a tutorial.

BACKGROUND OF THE INVENTION

Organizations and companies can amass tremendous amounts of information. However, transmitting this information in a useable format, including the right subset of the information and in the proper manner, to employees or individuals in the organization is often very difficult. In order to aid in this process, numerous computer-based training, tutoring and information delivery systems have been propagated to attempt to transfer data or information to users with no external input from a human (other than the user).

These training systems generally fall under the category of "computer-based training" (CBT), which includes "intelligent computer-based training" (ICBT), "intelligent computer-aided training" (ICAT) and "intelligent tutoring" (ITS) systems. These types of training systems share the objective of presenting instructional material to a human recipient (called a "user") using a computer program with little or no additional human intervention. These types of computer-based tutorial systems have been developed to reduce the cost associated with traditional classroom training (including travel costs, salaries for instructors, and excessive lost productivity during training).

In these computer-based training systems, the computer program must initially be provided with a database of the basic information from which it can compose the instructional material (e.g., text, graphics, charts, video clips, audio clips, etc.) that is provided or built by an "author." There are two distinct concepts surrounding the delivery of information using these tutorial systems that can be categorized as follows: (1) the content detailing the concepts to be conveyed to the user and (2) the manner in which the content is to be conveyed. For purposes of this disclosure, these two concepts are termed the "content" and "pedagogy" respectively.

Traditional CBT approaches generally include "authoring tools" that allow authors to develop and deliver on-line instruction programs. These prior art CBT systems focus on integrating numerous media formats into seamless, highly scripted presentations. These CBT authoring systems require the author to completely create and specify the pedagogy. Also the content and the pedagogy are highly interconnected within the database. Furthermore, these CBT systems have only been used to deliver knowledge to a student in a tutorial paradigm.

An extension of the tutorial systems can be generally categorized as a user-model tutorial. These user-model prior art tutorial systems attempt to develop models of the user's

2

state of knowledge prior to the tutorial. The emphasis of these approaches, including the overlay model, the bug library model, and the dynamic model computation, is on developing a student model and then using this model to determine choices in a predetermined pedagogy. However, these user-model approaches do not reduce any of the problems associated with creating the predetermined pedagogy by the author.

Another example of a computer aided training system is contained in U.S. Pat. No. 5,311,422, entitled "General Purpose Architecture for Intelligent Computer-Aided Training," which discloses a tool for building on-line training programs, including templates for content, pedagogy, and user model (i.e., model of the user's knowledge) that apply to a wide variety of domains. The tool can execute training programs using these general templates and cooperating expert system (rule-based) programs. However, these rule-based programs must be individually written for each new training application and therefore require a person skilled in the art of rule-based programming; a rare expertise when compared to traditional programming skills for languages such as C++ and JAVA. Furthermore, this patent still requires the author to create a fully-specified pedagogy (i.e., the pedagogy cannot be created automatically). Additionally, this prior art patent only discloses the delivery of knowledge to users in a tutorial paradigm (i.e., not in a help system, customer service, what-if scenario, etc.)

Another prior art training approach is virtual classroom training (VCT) which attempts to combine elements of classroom training with elements of CBT using video conference and other technologies. Typically, students are linked with an instructor in a virtual classroom using telecommunications technology and computer applications. While VCT offers some of the advantages of interaction among students and an instructor, it still has the drawback of requiring a human instructor. Additionally, these prior art VCT do not automatically generate pedagogies for the simple reason that the instructor determines how the material will be delivered. Furthermore, the initial implementation costs for VCT are extremely high and the VCT courses are not self-paced.

In all of the above-described prior art systems, the author must create and/or input a pedagogy for delivering the knowledge to the user. Therefore, during the building of the database, the author must also create the pedagogy (i.e., the manner in which the user will be delivered information from the database). In other words, the author must specify pedagogical details such as when content is to be displayed, how often it is to be displayed, whether it is optional or must be mastered prior to moving forward, etc. Furthermore, more flexible pedagogies are more desirable as they can address a greater variety of user backgrounds and skill levels, and can provide a given user with a greater variety of delivery interactions. In fact, some of these prior art tutoring systems incorporate very elaborate pedagogical specifications, including branching logic that can be based upon student inputs. However, another limitation of the prior art tutoring systems is that the greater the flexibility in the pedagogy, the greater the burden on the author who must input all the pedagogical details. This places a higher burden on either the author, the programmer creating the program, or both.

SUMMARY OF THE INVENTION

The present invention provides a dynamic knowledge creation and delivery system and method that substantially

eliminates or reduces disadvantages and problems associated with previously developed computer-based information delivery systems and methods.

More specifically, the present invention provides a system for that includes a knowledge delivery computer program stored in computer-readable form on a tangible storage medium for delivering concepts from a configuration data base to a user. An author can create the configuration data base using the knowledge delivery computer program so that the configuration includes a plurality of concepts and, optionally, a plurality of problems associated with the concepts. As the author creates the configuration, the knowledge delivery software allows the author to create a configuration taxonomy to define the relationships between the concepts. The knowledge delivery software can then automatically generate a pedagogy for the configuration based on the configuration taxonomy that defines how the concepts will be delivered to a user. The author may optionally create all or part of the pedagogy manually. The knowledge delivery software can also facilitate the delivery of the content within the configuration to a user according to the pedagogy. A user can access the configuration using the knowledge delivery software over the internet using a user interface stored on the client computer that allows the a user to send requests and receive responses over the internet.

The present invention provides an important technical advantage by separating the content to be delivered from the pedagogy or manner of delivery. Therefore, the present invention does not require any pedagogical input from the author creating the content knowledge database.

The present invention provides another technical advantage by automatically creating the pedagogy for delivering knowledge to a user. The automatic generation of the pedagogy is based on the content of the knowledge base and the taxonomy of, or relationships between, those concepts.

The present invention provides still another technical advantage by automatically generating different pedagogies from a single concept knowledge base.

The present invention provides yet another technical advantage by allowing the delivery of content from the knowledge data base in a variety of different formats using a variety of different delivery mechanisms (including the internet).

The present invention provides another technical advantage by allowing the author of the knowledge base to optionally override the automatic pedagogy mechanism to create a pedagogy that the author specifies.

The present invention provides a technical advantage because it can deliver knowledge to a user using a general architecture that is adaptable for a variety of instructional fields in addition to the traditional tutorial paradigm (e.g., in a help system environment, a customer service scenario, a what-if scenario, etc.).

The present invention provides still another technical advantage by being adaptable to create multiple concept knowledge bases, to create multiple configurations, and to provide knowledge delivery to multiple users at a time from any of multiple configurations.

The present invention provides another technical advantage by allowing for the creation and integration of a user model for every user/configuration pair that includes a history of all the elements of the given configuration presented by the invention to the user, as well as a record of the user's response to each element.

The present invention provides another technical advantage in that it does not require either the author or the user

to have an understanding of neural network programming, artificial intelligence programming, or even more common programming languages such as C++ or JAVA. The present invention utilizes browser extension technology to allow both the author and the user to "point and click" to execute functionality.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following description taken in conjunction with the accompanying drawings in which like reference numerals indicate like features and wherein:

FIG. 1 is a diagrammatical representation of a client-server architecture incorporating one embodiment of the knowledge delivery program of the present invention;

FIG. 2 shows the interaction between an author and the configuration editor and a number of users and the tutor maker;

FIG. 3 illustrates in more detail the subprogram of the configuration editor and tutor maker of FIG. 2;

FIG. 4 illustrates in more detail the configuration editor of the FIG. 2 embodiment;

FIG. 5 illustrates in more detail the tutor maker of the FIG. 2 embodiment;

FIG. 6 shows one embodiment of the user interface according to the present invention;

FIG. 7 illustrates in greater detail the interaction between the user and the tutor maker;

FIGS. 8-14 are schematic representations of the data structures for programming elements of the present invention for an object oriented embodiment;

FIGS. 15-27 show embodiments of various windows that can be used to implement the knowledge delivery program of the present invention;

FIG. 28 is a flow diagram illustrating the operation of one embodiment of the present invention;

FIG. 29 is a flow diagram illustrating the operation of the configuration editor;

FIG. 30 is a flow diagram illustrating the operation of the concept editor;

FIG. 31 is a flow diagram illustrating the operation of the add/update concept function of the present invention;

FIG. 32 is a flow diagram illustrating the operation of the create concept function of the present invention;

FIG. 33 is a flow diagram illustrating the operation of the modify concept function of the present invention;

FIG. 34 is a flow diagram illustrating the operation of the problem editor function of the present invention;

FIG. 35 is a flow diagram illustrating the operation of the update problem function of the present invention;

FIG. 36 is a flow diagram illustrating the operation of the pedagogy editor function of the present invention;

FIG. 37A is a flow diagram illustrating the operation of the pedagogy generator function of the present invention;

FIG. 37B shows a window illustrating an automatically generated pedagogy for a sample configuration;

FIG. 37C shows a window containing another pedagogy automatically generated according to FIG. 37A;

FIG. 38 is a flow diagram illustrating the operation of the delete pedagogy function of the present invention;

FIG. 39 is a flow diagram illustrating the operation of the add/update pedagogy function of the present invention;

FIG. 40 is a flow diagram illustrating the operation of the tutor maker according to the present invention;

FIG. 41 is a flow diagram illustrating the operation of the tutor maker action handler function of the present invention;

FIG. 42 is a flow diagram illustrating the operation of the load configuration function of the present invention;

FIG. 43 is a flow diagram illustrating the operation of the load knowledge base function of the present invention;

FIG. 44 is a flow diagram illustrating the operation of the delete configuration function of the present invention;

FIG. 45 is a flow diagram illustrating the operation of the delete knowledge base function of the present invention;

FIG. 46 is a flow diagram illustrating the operation of the generate HTML function of the present invention;

FIG. 47 is a flow diagram illustrating the operation of the toolbar HTML generation function of the present invention;

FIG. 48 is a flow diagram illustrating the operation of the event handler function of the present invention;

FIG. 49 is a flow diagram illustrating the operation of the pop-up handler function of the present invention; and

FIG. 50 is a flow diagram illustrating the operation of the mode handler function of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiments of the present invention are illustrated in the FIGURES, like numerals being used to refer to like and corresponding parts of the various drawings.

The present invention includes software programming to deliver knowledge to users. The knowledge delivery can occur using a variety of mechanisms including a CD-ROM on a stand-alone computer, or over an intranet or the internet. The description of the present invention will presume a client-server architecture and delivery over the internet, though any other computer-based delivery mechanism can be employed. The dynamic knowledge delivery program 100 of the present invention includes two components (each of which can be a stand alone software program): (1) software to produce a configuration and (2) the software to deliver knowledge from the configuration to the user.

For purposes of the present invention, a distinction will be drawn to two categories of information within the configuration: the "content" and the "pedagogy." The content represents the concepts to be conveyed to the user (i.e., what information will be provided the user during instruction). The pedagogy is the manner in which the information is to be displayed to the user (i.e., how the instruction will be performed). The dynamic knowledge delivery program 100 includes an automated pedagogy generator that automatically determines, based on taxonomy input by the author during the building of the configuration, how the knowledge should be delivered to the user without additional input from the author. In this way, the present invention completely separates content from pedagogy. Thus, the author does not have to develop both the content and the manner of delivery, but rather must just develop the content and the relationships among the concepts that make up the content. The manner of delivery, or the pedagogy, can then be automatically generated by the present invention. Another distinction can be drawn between categories of information contained with the "content" portion of the configuration 10: the "descriptive content" and the "referential content". The descriptive content is the actual text, video, sound, etc. that describes a component of the configuration. For example, the description of a particular concept might include a drawing or figure

and accompanying text. Also, a description of a test question might consist of the text for the statement of the question, the text for each multiple choice answer, and the text to explain the correct answer to the question. The referential content is the linkage (e.g., pointers) to the description content plus any relational linkages among elements of the referential content. Thus, a concept would refer to the descriptive content for that concept, and it would refer to any examples or test questions related to that concept. The taxonomy mentioned previously is part of the referential content of a configuration 10.

The present invention includes dynamic knowledge delivery computer program 100 executed within a computer system 200, such as that shown in FIG. 1. Computer system 200 is a client/server architecture that includes the internet, and includes client computer 220, server computer 230, and network computer 240. Client computer 220 connects to server computer 230 over network 210, while network computer 240 connects to server computer 230 over network 210. It should be understood that network 210 can be the same network or a different network in practice (e.g., the network connecting client computer 220 to server computer 230 could be the internet, while the network connecting the network computer 240 to the server computer 230 could be a company intranet). It should be further understood that a variety of other computer systems could be employed, including a single computer executing all of the client, server and network side functions of FIG. 1.

Each computer in FIG. 1 includes a central processing unit (CPU) or processor 12 and a storage device or memory 14. Dynamic knowledge delivery computer program 100 includes the configuration editor program 110, the tutor maker program 120, and the user interface 130, each of which can reside in memory 14 (for example, random access memory or RAM) within the server computer, the network computer, and the client computer, respectively. As shown, configuration 10 can reside in file memory 15 (e.g., hard drive, floppy disk, etc.) that resides on or is accessible by server computer 230. The dynamic knowledge delivery computer program 100 includes instructions that, when executed, cause the CPUs 12 to perform the functions described herein. In another embodiment each of the three programs could reside on a single computer that an author could access to create the configuration 10 and a user could then access in order to deliver knowledge from the configuration 10. In yet another embodiment, dynamic knowledge delivery computer program 100 is stored in a computer readable medium of a removable computer disk, such as a floppy disk or compact disk.

For purposes of the present invention, the configuration building software is referred to herein as the configuration editor 110, while the knowledge delivery software is referred to as the tutor maker 120. An internet architecture such as that shown in FIG. 1 will also include a user interface 130, which is shown to include a browser 28 and a browser extension 30. Configuration editor 110 can include five sub-programs: the concept editor 16, problem editor 18, pedagogy editor 20, tutor manager 44, and pedagogy generator 50, each of which will be described in more detail. It should be noted that the pedagogy generator 50 can optionally be located within the tutor maker 120, however locating the pedagogy generator 50 within the configuration editor 110 allows an author to make pedagogical input as an alternative to the automated pedagogy generation of the pedagogy generator 50. Tutor maker 120 can include three sub-programs, tutor manager 44, pedagogy generator 50, and HTML generator 36. In both configuration editor 110

and tutor maker 120, a tutor manager 44 contains the concept reader 22, the problem reader 24, and the pedagogy reader 26. FIG. 1 also shows configuration 10 loaded within file memory 16 on server computer 230.

As illustrated in FIG. 2, the interactions between the dynamic knowledge delivery program 100 and human beings occur in two phases: the first phase between an "author" 32 and the program 100 to develop the configuration 10, and the second phase between a "user" 34 and the program 100 to deliver content from the configuration 10. The configuration editor 110 allows the author 32 to develop configurations 10 from which content will be extracted for delivery to user(s) 34. The tutor maker 120 allows a user 34 (or multiple users 34 simultaneously) to have content delivered to them according to the pedagogy that can be automatically generated by pedagogy generator 50. FIG. 2 shows that any number of data configurations 10 can be built author(s) 32 from the configuration editor 110 and/or accessed by user(s) 34 from the tutor maker 120. Each configuration can be completely independent from every other configuration. FIG. 2 also illustrates that any X number of users 34 can access any N number of configurations 10 where X does not necessarily equal N. Thus, two or more students 34 can access the same configuration 10 simultaneously using concurrent programming techniques.

Configuration 10 is the underlying data structure that consists of a concept knowledge base, and optionally a pedagogy knowledge base and a problem knowledge base. Different configurations 10 may be constructed by mixing together different concept knowledge bases, problem knowledge bases, and pedagogy knowledge bases (e.g., the same concept knowledge base can be used in multiple different configurations and multiple different tutorials). The knowledge bases are the data bases of information or "concepts" from which certain of the concepts are chosen and delivered to user(s) 34. The term "knowledge base" is used to differentiate this data base from typical informational databases because the knowledge base incorporates a taxonomy or dependency of relationships among the concepts.

For example, a broad concept that could be included within a knowledge base could be a computer. The concept of a computer also includes sub-assembly concepts such as a keyboard, a processor, a display screen, a memory, and component concepts such as transistors, resistors, and microchips. The computer can also include concepts such as software programs and architectures such as an intranet or internet. Further concepts can include how the computer actually works using the various sub-assemblies, software and architectures. Depending on the user, different concepts from the knowledge base can be delivered using the dynamic knowledge delivery program 100 of the present invention. For a sales person, concepts such as the operation of a microprocessor to execute a program might represent too much detail, however, that might be exactly the level of detail required for an electronic technician building the computer. Thus, the dynamic knowledge delivery program 100 allows author 32 to build knowledge bases in such a manner as to allow different format and content delivery to different users 34.

One of the unique aspects of the dynamic knowledge delivery software program 100 is that it allows the author 32 to enter into the configuration 10 not only the concepts the author 32 wants to teach or transmit to the user(s) 34, but also the taxonomy of the concepts or the relationship among those concepts. The dynamic knowledge delivery software program 100 then leverages that taxonomy with software programming to automatically build a pedagogy based on the taxonomy.

FIG. 3 shows a more detailed block diagram of the system shown in FIG. 1. As shown in FIG. 3, the author 32 accesses the configuration editor 110 that includes concept editor 16, which is used by the author 32 to make the concept knowledge base 11, problem editor 18, which can be used to make the problem knowledge base 13, and pedagogy editor 20, which can be used to make the pedagogy knowledge base 15, and these three knowledge bases comprise configuration 10. The tutorial manager 41 within configuration editor 110 is connected to tutorial manager 44 contained within tutor maker 120 (each of which contains the concept reader 22 (shown as part of concept editor 16), problem reader 24 (shown as part of problem editor 18), and pedagogy reader 26 (shown as part of pedagogy editor 26)). As shown in FIG. 3, tutor maker 120 includes four sub-programs which are the action handler 42, HTML generator 36, tutorial manager 44, and user session 48. Tutor maker 120 also includes tutorial database 46 and can optionally access user model 52 which is a data base modeling the user 34. The user 34 accesses tutor maker 120 over network 210 using the user interface 130 which includes, for an internet application, a browser extension 30 and standard browser 28 (such as internet Explorer, Netscape Navigator or Netscape Communicator).

Configuration editor 110 facilitates the building of configuration(s) 10. As shown in FIG. 3, configuration 10 includes a concept knowledge base 11 component, a problem knowledge base 13 component, and a pedagogy knowledge base 15 component. The concept knowledge base 11 is the portion of configuration 10 that includes the concepts to be delivered to a user 34. The concept editor 16 of configuration editor 110 is the subprogram utilized by the author 32 to input concepts into a concept knowledge base 11 and to create the hierarchy of relationships among concepts known as the taxonomy. Thus, as an author creates a knowledge base 11, the taxonomy is created as well (i.e., as the author 34 designates the relationships among the concepts). The combination of the taxonomy and automated pedagogy generation provide an extremely flexible knowledge delivery system that does not require pre-scripting the way the information will be delivered. The descriptive content for the concepts can be created using any number of data formats (e.g., text, tables, audio, and video of all types) so long as it can be converted into a format readable by the dynamic knowledge delivery program 100. For example, an internet client-server architecture as shown in FIG. 1 currently requires the descriptive content developed by the author 34 using the concept editor 16 to be compatible with or convertible to HTML. This allows the delivery of the content to users 34 over the internet. As the author completes the inclusion of data into the concept using the concept editor 16, a pointer is created that will point to all of the data associated with the concept.

Furthermore, each concept is incorporated into a structure of relationships, or taxonomy, based on each concept's relationship to previously created concepts (i.e., the "referential content"). For example, suppose the concept to be learned by the user is asset allocation. The base concept would be asset allocation which would include a number of concepts that need to be taught in order to understand asset allocation. Assume that at least the concepts "time-risk tradeoff", "stocks", "bonds", "investment pools", "investment goals", "dollar cost averaging", "value averaging" and "risk tolerance" must be delivered to the user to understand asset allocation. The author 32 will use the concept editor 16 to create descriptive content for each of these concepts by placing information (again, text, video and audio) within each concept. In an internet client-server environment, each

concept would have HTML pages associated with it that contained the content. Further assume that the author 32 thinks that in order to understand "time-risk tradeoff", a person must first understand "investment goals", "risk tolerance", "dollar cost averaging" and "value averaging". Therefore, the taxonomy for the concept of "time-risk tradeoff" is that this concept depends on the "sub" concepts of "investment goals", "risk tolerance", "dollar cost averaging" and "value averaging". Each concept, as it is created by the author 32, would receive a similar taxonomical indicator among the existing concepts. Further assume that "dollar cost averaging" is a leaf concept, or a concept that does not require any further explanation. Thus, no concepts will be designated as supporting concepts for the concept of dollar cost averaging in the taxonomy for asset allocation. However, if so desired by the author, dollar cost averaging could further depend on the concepts of "currency" and "time".

The author may use problem editor 18 to build a problem knowledge base 13 and the pedagogy editor 20 to create a pedagogy knowledge base 15. It should be noted that the both the problem knowledge base 13 and the pedagogy knowledge base 15 are optional. The pedagogy editor 20 allows the author 32 to create some or all of the pedagogy (i.e., manner of delivery of content) based on the author's experience. However, this is entirely optional as the pedagogy generator 50 will automatically create a pedagogy based on the taxonomy of concepts input by the author 32. For example, if the author 32 has a goal to teach asset allocation, the pedagogy generator 50 can automatically generate a tutorial for asset allocation based on the taxonomy. The pedagogy editor 20 allows the author 32 to create the pedagogy independently of the pedagogy generator 50 or to modify the a pedagogy that was automatically generated.

The author 32 may also create a problem knowledge base 13 using problem editor 18. A problem knowledge base 13 would include examples and questions (collectively called "problems") relating to the concepts in a concept knowledge base 11. As with the concept knowledge base, the problem knowledge base was both descriptive and referential content (wherein descriptive is the HTML for problems and referential content is the links to concepts. For example, the author 32 could have concepts in a tutorial delivered with examples as the tutorial is delivered. In a tutorial setting, an author 32 may want to test the user 34 (e.g., use of the program 100 to re-certify engineers). Thus, the author 32 could build a problem knowledge base 13 of questions using problem editor 18. In an alternative embodiment, the problem knowledge base 13 could include a simulator program that automatically generates the descriptive content for questions and/or examples related to the concepts based on the concepts input by the author 32. An example of such a program that could automatically generate problems is disclosed in *Automatic Student Modeling and Bug Library Construction Using Theory Refinement*, Baffes, Paul, (December 1994), which is incorporated by reference herein in its entirety.

Each of the concept editor 16, problem editor 18 and pedagogy editor 20 can be used to later modify existing configuration 10 by adding, deleting or modifying concepts, problems, or the pedagogy. The existing concepts, problems, and pedagogy within the concept knowledge base 11, problem knowledge base 13 and pedagogy knowledge base 15 can be accessed by the author 32 using the concept reader 22, problem reader 24, and pedagogy reader 26, respectively. In the preferred embodiment, the referential content

for each concept, problem, and pedagogical step within the respective knowledge bases will be software objects that are linked to other software objects. Programming the concepts, problems and pedagogical steps as objects in an object-oriented environment can simplify usage by multiple users.

For a client-server architecture incorporating a browser 28 within the user interface 130, the descriptive content each of the concept, problem, and pedagogy knowledge bases would be written in or converted to HTML pages. For example, each concept in a taxonomy will have one or more different content HTML files associated with it, and optionally there may be problem, example and/or question HTML pages and pedagogy HTML pages associated with each concept.

In one embodiment, each configuration 10 and all of the descriptive content files associated with the concept knowledge base 11, problem knowledge base 13 and pedagogy knowledge base 15 that comprise the configuration 10 are built under a single file directory to simplify bundling the configuration 10 and moving it (e.g., to place it on a disk). This single directory also facilitates delivery of the content of the configuration 10 over the internet because one can easily relocate the top-level directory from the network computer 240 to the server computer 230. Once the referential content for the concept knowledge base, problem knowledge base, and pedagogy knowledge base of a configuration is loaded into RAM memory on the server computer by the tutorial manager, the descriptive content files for the configuration can be readily accessed.

Tutor maker 120 facilitates the delivery of content from the configuration 10 to the user 34. The action handler program 42 will receive a request from a user 34 via browser extension 30 (e.g., log in) and will either process the request through the tutorial manager program 44 or the user session 48. A user 34 will log in and request a particular topic (i.e., configuration 10). The tutorial manager 44 will track whether the referential content of the requested configuration 10 has already been loaded from file memory 15 into RAM memory 14 by another user 34, and if so, will not reload the topic, but will process the request from the loaded version of the configuration. The tutorial manager 44 tracks all of the pieces of configuration 10 to ensure that the tutorial manager program 44 only loads the referential content of a particular knowledge base, problem base, or pedagogy base one time. In this way, the time delay due to file access is minimized. Thus, a previously loaded concept knowledge base 11 in use by another user 34 need not be reloaded when a new request for the same concept knowledge base 11 is received at the tutorial manager 44. Thus, the tutorial manager 44 reads the configuration 10 resident in file memory 15 in the server computer 230 and loads into RAM memory 14 the referential content of at least the concept knowledge base 11 of configuration 10 which is then used for all users 34 accessing this particular concept knowledge base 11. All loaded files in the tutorial manager 44 are maintained in the tutorial database 46, which is essentially a collection of pointers pointing to the HTML pages of the configuration. Thus, if multiple users 34 are viewing the same configuration 10, the users are accessing the configuration 10 at the tutorial data base 46, rather than directly accessing configuration 10. The tutorial database 46 is the resident set of all configurations 10 currently being viewed by users 34.

The user session 48 establishes a thread for each user 34. In one embodiment, the tutor maker 120 is a single program that runs on the network that can service any number of different users 34 simultaneously. In order to keep track of the different users 34, the tutor maker 120 uses parallel the

standard computer science concept known as "threads". A thread is the software abstraction for running a separate process which allows multiple processes to be running literally simultaneously either on a true parallel computing machine or a standard computing platform which simulates parallel processing (most modern operating systems have such simulation). A thread is created for each user 34 that logs into the tutor maker 120. This creation of a separate thread for each user 34 allows for synchronizing actions among multiple users. The user session 48 allows synchronizing of threads to prevent loss of commands or erroneous handling of commands the user 34 inputs.

Also shown in FIG. 3 is an optional user model 52 data base that can be accessed by the user session 48. The user model 52 is built by the user session 48 and is a historical data base of a user's interactions with the tutor maker 120. The user model 52 could also incorporate much more complex user modeling techniques that have been developed to more precisely deliver content to the user 34.

Each action from the user 34 goes to either the tutorial manager 44 or to the user session 48 if a user session has already been established. If a user session has not been established, the tutorial manager 44 either loads the configuration 10 requested by user, or if it is already loaded, accesses it in the tutorial database 46. A new user session is then established and associated with the loaded or retrieval configuration, and the tutorial manager records this new association with the configuration in the tutorial database. Responses from user session 48 (based on requests from the user 34) will be put through the HTML generator 36 to place the response in HTML format for transmission back through the internet to the browser 28 where the user 34 receives the response. At every exchange, the user 34 makes a request and the tutor maker 120 responds with the HTML. The HTML generator 36 can be a separate program or file in order to ease the incorporation of changes in HTML versions.

FIG. 3 shows automatic pedagogy generator 50 resident in the pedagogy editor 20 within configuration editor 110 and resident in the HTML generator 36 within the tutor maker 120. However, if the pedagogy generator 50 only resides in the tutor maker 120, the author 32 will have no access to it. Putting pedagogy generator 50 in configuration editor 110 allows the pedagogy to be automatically generated by pedagogy generator 50 and additionally modified, if so desired, by the author 32.

The tutor maker 120 allows the presentation or delivery of content in different formats as chosen by the user 34. Initially, the user will log in and the tutor maker 120 will provide the user a list of topics (i.e., configurations 10). Upon choosing a topic, the user 34 is given a set of options on presentation style (e.g., standard tutorial, a question and answer session, a review, a surf/search session, etc.) In one embodiment, a custom tutorial could be presented by the tutor maker 120 based on the user's answers to a test covering the topic of choice.

The tutor maker 120 allows a user 34 to have content from the configuration 10 delivered based on the pedagogy created by author 32. The concepts can include hyperlinks to allow a user 34 to see particular content even when the pedagogy would not automatically deliver that particular content to the user 34. In addition, a user 34 could ask for examples on demand. The tutor maker 120 can track which examples a user 34 has seen and only show new examples, or recycle the examples based on the oldest viewed one. The user 34 can also request a test, whereupon the tutorial

manager 44 will extract one or more test questions from the problem knowledge base 13. The tutor maker 120 can score the test immediately and provide feedback on incorrect answers by calling content related to the question from the concept knowledge base 11 or the problem knowledge base 13. Furthermore, if the user incorrectly answers a question, the tutor maker 120 can generate the related concepts and a taxonomy for that particular question. Thus, the tutor maker 120 can either progress through the topic using the standard pedagogy generated by pedagogy generator 50 or created by the author 32, or alternatively can be directed by the user 34 through the topic in an entirely user-driven manner.

FIG. 4 is a diagram illustrating the overlap of the configuration editor 110 and the tutor maker 120. As indicated earlier, both the configuration editor 110 on the network computer 240 and the tutor maker 120 on the server computer 230 contain a concept reader 22, a problem reader 24 and a pedagogy reader 26 (note also that the pedagogy generator 50 can also be part of both programs, as explained previously). This allows both the configuration editor 110 and the tutor maker 120 to load a configuration 10 using the readers. In other words, in order for an author 32 to modify an existing configuration 10, the author 32 must also read it via the concept reader 22; likewise, if the configuration 10 will be presented to the user 34, it must also be read. To illustrate, assume an author 32 must edit the problem knowledge base by adding a question and modifying an example. Using the configuration editor 110, the author 32 would invoke the problem reader 24 to read the referential content for the desired problem knowledge base 13. The author 32 could then use the problem editor 18 either to modify description content for the desired example or to modify the referential content for that example (e.g., associate the example with a different concept). When the user 34 then accesses the content for this example, it is accomplished via a call to the tutor maker 120 which invokes the action handler 42 which invokes the tutorial manager 44 which will use the problem reader 24 to load the referential content for the problem knowledge base through which the descriptive content for the example is retrieved and sent back to the user. Again note that if the user has already established a user session 48 via a previous input, then the user session itself has a reference to the referential content for the problem knowledge base to gain access to the descriptive content for the example.

The remainder of FIG. 4 shows the data (i.e., content) contained in the various knowledge bases. The configuration 10 comprises a concept knowledge base 11, a problem knowledge base 13 (made up of question knowledge base 17 and example knowledge base 19) and a pedagogy knowledge base 15. The author uses the concept editor 16 to create/edit the concept knowledge base 11, the problem editor 18 to create/edit the problem knowledge base 13 and the pedagogy editor 20 to create/edit a pedagogy in pedagogy knowledge base 15. To better illustrate the configurations 10, presume that an author 32 has created one concept knowledge base 11, 25 different problem knowledge bases 13, and three different pedagogies in pedagogy knowledge base 15 (e.g., a pedagogy for beginning, intermediate and advance users). Each triplet of the concept knowledge base, one problem knowledge base and one pedagogy would constitute a single configuration 10.

FIG. 5 is a diagram detailing the tutor maker 120 of FIG. 3. As shown, when the tutorial manager 44 builds a representation of configuration 10 as tutorial database 46, concept reader 16 accesses the appropriate concept knowledge base 11, problem reader 16 accesses the appropriate problem

knowledge base 13 and pedagogy reader 15 accesses pedagogy knowledge base 15. FIG. 5 further illustrates the various pedagogy options open to user 34. If user 34 simply follows the existing pedagogy created by the configuration editor 120, the user session 48 returns responses to the user through the standard HTML generator 37 as shown in path labeled (a) within HTML generator 36. In contrast, if user 34 decides to deviate from the existing pedagogy, the user session 48 returns responses to the user through the pedagogy generator 50, then standard HTML generator 37 as shown in path labeled (b) within HTML generator 36. Finally, the user 34 may opt to manually traverse the content of a configuration in which case the pedagogy generator is also bypassed via path (a).

FIG. 6 is a diagram illustrating the interaction between the user 34 and the tutor maker 120 via user interface 130. User interface 130 comprises a browser 28 and a browser extension 30 as shown in FIG. 6. Browser 28 can be any internet browser, such as Microsoft internet Explorer or Netscape Navigator or Communicator. It should be understood that different user interfaces 130 could be developed that do not require the use of an internet browser. However, for delivery of content to a user 34 over the internet, the use of a browser 28 within user interface 130 is the preferred embodiment. The user interface 130 extends the browser 28 with browser extension program 30 that includes an event handler 60. An event handler 60 is a well understood computer science mechanism for allowing a user 34 to generate an event to be acted upon by another program (in this case the tutor maker 120). The particular event handler 60 shown in FIG. 6 includes a pop-up handler sub-program 62, a mode change sub-program 64 and an action sender sub-program 66. As user 34 makes a request, that request goes to the event handler 60. The three sub-programs receive the input commands from the event handler 60 and if either a mode change or a window pop-up is required for that command, the pop-up window 62 or mode changer 64 respond accordingly. The action sender 66 will transmit the command through the network 210 to the action handler 42 of the tutor maker 120. An event can either generate an action that is sent to the tutor maker 120 by action sender 66 or the event can build directly into the browser (e.g., close a window type event).

The browser extension 30 can be built to be compatible with both currently dominant internet browsers, Microsoft internet Explorer and Netscape Navigator or Communicator, by writing the browser extension 30 in javascript, jscript, or any other compatible browser scripting language. However, it should be understood that the user interface 130 could be run with any browser by, for example, including a step for detecting which browser 28 was running on the user's machine and sending a compatible browser extension 30 for that particular browser 28. During the operation of one embodiment of the dynamic knowledge delivery program 100 of the present invention, each time the user 34 inputs a command, the browser extension 30 is loaded (i.e., the particular Java script file that is the browser extension 30). The browser extension 30 includes a set of functions that sit within browser 28. Thus, whenever the user 34 inputs a command, for example clicks on a hyperlink, the browser extension 30 is called and every HTML Web page that is generated by the tutor maker 120 and sent back to the browser 28 includes a reference to the browser extension 30. For example, the top line of the generated HTML page can include a reference to the browser extension 30, and when the browser 30 receives the HTML page from the tutor maker 120, the browser 28 now knows how to interpret the HTML page and reads it.

The dynamic knowledge delivery program 100 of the present invention can be programmed using C++, JAVA, or any other commonly utilized computer programming language. In the preferred embodiment for use over the internet, the programming language is JAVA to facilitate the delivery of content to a user over the internet. Another novel feature of the present invention includes, in an internet environment, the ability to access the tutor maker 120 using browser extension 30 and retrieve responses through browser 28 without actually calling different URL Web pages. This is accomplished using "script call" where the call actually sends the request for a URL over the internet to a servlet program that takes the request, generates a response in HTML format and sends the HTML page to the user's existing URL. Thus, the tutor maker 120 need not have every potential HTML file that could be a response to a user request existing as a Web page with an associated URL, but rather each HTML page response is generated upon each request and sent back to the user via the browser 28. For example, when the user 34 logs in, a call goes to the tutor maker program 120 to generate and send back a particular HTML page that is a form that can include a login box, a password box, and a submit button. Similarly, user 34 can call the tutor maker 120 on the server computer 230 to generate all of the HTML page responses upon request.

This novel architecture also allows for delivery of a configuration 10 to a particular user 34 in a unique manner compared to typical internet applications. Assume a particular user 34 logs in and is having a particular tutorial delivered. Further assume the user leaves the computer, but remains logged on at a certain point in the tutorial. The user 34 now goes to another computer and links to the internet using a different browser 28 (e.g., uses Netscape Navigator on the second computer while the first was using Microsoft Explorer). The user 34 again logs into the tutor maker 120 on server computer 230 to view the same tutorial. The tutor maker 120 according to the present invention can determine that this user 34 is already viewing this tutorial in a different place on the internet and can send a message asking the user whether the user wants to log off the original computer and view the tutorial from the present location. If the user 34 responds in the affirmative, the tutor maker 120 will log off the first location and bring up the tutorial in precisely the place the user left off earlier. Thus, the dynamic knowledge delivery program 100 of the present invention is internet aware in that it can (1) allow multiple users to simultaneously view the same or different configurations and (2) can move the same user from a first location to a second location while maintaining the history on a particular tutorial (without the user having to save the tutorial on a disk and take it to the new physical location). This particular novel programming feature can be used in a variety of information delivery processes over the internet. For example, in an e-commerce application, a shopper could be half way through a transaction and need to change locations. The shopper could leave the first computer with the transaction half completed, log on to the same site at a different computer and continue the transaction at the same place. Essentially, this programming feature will work for any internet function that requires a user to log on.

FIG. 7 is a partial detailed diagram of the tutor maker 120 of FIG. 3 to illustrate the ability of the present invention to manage multiple users 34 viewing multiple configurations 10. In FIG. 7, "x" users 34 have opened user sessions 48 and have accessed "n" configurations 10. Tutorial manager 44 has therefore labeled "n" configurations 10 into the tutorial data base 46 (i.e., by reading in the three knowledge bases

of each configuration 10). Thus, x users 34 are accessing the n tutorial data bases 46. For example, assume ten total users and that user number 5, user number 7 and user number 10 are accessing the same concept knowledge base 11. Therefore, tutorial database numbers 5, 7 and 10 share the same concept knowledge base. However, only one instance of that particular concept knowledge base would be loaded in memory within tutorial data base 46. Let's further assume that user numbers 5 and 7 are also accessing the same problem knowledge base, but user number 10 is accessing a different problem knowledge base. An instance of each of these separate problem knowledge bases would then be loaded in tutorial data base 46. Thus, for users 5, 7 and 10, we have three user sessions 48, two tutorial databases 46 each sharing one instance of the concept knowledge base.

FIGS. 8-14 illustrate exemplary abstract views of the data structures for some of the components of the dynamic knowledge delivery program 100. It should be understood that the elements could include different data structures than those exemplary data structures shown. FIGS. 8-14 show the abstract views of objects developed using object-oriented programming.

FIG. 8 shows a concept knowledge base data structure 68, which is the referential component of a concept class. The concept knowledge base is the set of concepts created by the author that will be delivered to the user. The set of concepts can be stored in a variety of standard computer science mechanisms (e.g., lists, hash tables). The concept knowledge base data structure 68 shown in FIG. 8 includes a name, a title, dependent links, supported links, views and problems links. The name is the name the author assigns the concept and the title is the way the concept appears when shown to the user (which can be different from the name). The dependent links are the references to additional concepts that this concept breaks down into, while the supported links are references to the additional concepts that this concept supports. The dependent and supported concept links represent the taxonomy created by the author when inputting the concepts. The problem links are the set of examples or questions associated with this concept, and are used to link the concept to all of its associated problems in the problem knowledge base. The views would contain references to the descriptive content associated with the concept that may be presented to the user. This descriptive content may include audio chips, text, streaming video or whatever other form of data represents the concept description. The concept view 70 is an instance of the concept class that is further illustrated to have an HTML reference (for example, a detailed view for a beginner versus a more limited view for an advanced user), and a type that is a label associated with the particular kind of view (detailed versus limited).

FIG. 9 shows a problem knowledge base data structure 72, which is a problem class. Again, the problem knowledge base is a set of problems that can be stored as a hash table, list, etc. FIG. 9 also includes the inheritance hierarchy of the problem class to illustrate that other objects are derived from the abstract problem class 72. As problem inheritance hierarchy 74 illustrates, an abstract problem type is either an example or a test question, where the test question is again an abstract class that is either an essay, a multiple choice question, or a true/false question. Thus, the derived classes of problem, essay, multiple choice question and true/false question are derived from the general problem class and "inherit" all of the attributes of problem class 72, including a name, concept reference, HTML reference, and explanation. Thus, each problem has some kind of name associated with it, a pointer to the concept with which it's associated,

HTML that describes it, and an explanation of the problem. The explanation data structure 76 is an instance of the problem class 72 that provides context for the explanation and an HTML reference. The multiple choice question data structure 78 is an instance of the problem class 72 that includes an answer list and the correct answer, where the correct answer data structure 80 includes a problem reference (what problem is this the correct answer for), an HTML reference (the answer) and a type (what type of answer).

FIG. 10 illustrates the abstract data structures and inheritance hierarchy for the pedagogy knowledge base. The pedagogy knowledge base inheritance hierarchy 82 shows the abstract pedagogy element class from which the pedagogy concept, pedagogy unit and pedagogy test are derived. Thus, the pedagogy element actually is a concept, a unit (i.e., a group of concepts) or a test. The pedagogy element at the start of the pedagogy is called the pedagogy root data structure 84 that has a name and it's children. Note that every pedagogy element has a reference to its parent, which would be empty if the pedagogy element was the root. For example, Concept 1 could have a parent of Chapter 1, whereas Chapter 1 may have no parent. The pedagogy element 86 is the abstract class from which the pedagogy concept instance 88, pedagogy unit instance 92 and the pedagogy test instance 90 are derived. Therefore, pedagogy concept class 88, pedagogy unit instance 92 and the pedagogy test instance 90 inherits everything from the pedagogy element class 86, i.e., the parent, plus the concept link and the view type. The concept link indicates the location of the concept within the pedagogy and the view type indicates the type of descriptive content to be retrieved from the concept to be presented to the user. Similarly, the pedagogy unit instance 92 would include the name for the unit, a reference to the children concepts contained in the unit (e.g., sections contained in a chapter), and any descriptive content (HTML) associated with the unit as a whole (e.g., an introduction to a chapter). The pedagogy test instance 90 includes the size (e.g., number of questions) and the list of concepts that are covered by the test.

Similarly, FIG. 11 shows a configuration data structure 94 that can include a name and references to the concept knowledge base, the problem knowledge base and the pedagogy knowledge base. Note that the references to the problem knowledge base and pedagogy knowledge base can be optional. FIG. 12 shows exemplary tutorial database data structure 96, shared configuration data structure 97 and shared tutorial object data structure 98. Because the tutorial database tracks the concept knowledge bases, problem knowledge bases and pedagogy knowledge bases, the tutorial data base structure 96 can therefore have a shared concept knowledge base link, shared problem knowledge base link, a shared pedagogy knowledge base link, and a shared configuration link. The shared tutorial object 98 is an abstract class that can work for any shared knowledge base. Thus, the shared concept, problem, and pedagogy knowledge bases can all include an object reference, valid flag and a count. If the count goes to zero (i.e., no instances of a shared object are open), the object can be closed. The next time a user wishes to access that object, it must be loaded into the tutorial database 46. Similarly, the shared configuration can include all of the properties of the shared configuration abstract class 97, including a name, the triplet comprising the configuration, a valid flag, and a count.

FIG. 13 shows exemplary user session data structure 102 and the user data structure 104 associated with it. As shown, the user session 104 includes an expiration time, a session ID, and a reference to the associated user data structure 104

17

which has a user ID, a user model reference and a shared configuration reference. The session ID is the unique ID assigned to that particular session. This unique ID enables the program 100 to determine that a request from a user is associated with a particular session. The user reference includes the information about the user, including in this example the user's ID, any user model that has been built for this user, and any shared configuration the user is viewing. FIG. 14 shows a user model data structure 106 which contains lists of user model element data structures 108 (that includes a count, the last time viewed, and a name). As elements of the configuration are displayed to the user, user model element data structures are created and stored in the user model 106. The user model data structure 106 further includes any historical information about the user that is recorded by the dynamic knowledge delivery program 100 (exemplary data shown in FIG. 14). The saved test instance 112 of the user model class 106 can be used to save previous test scores of a user, and as shown includes the question links and the answer array.

FIGS. 15-27 show exemplary windows of the dynamic knowledge delivery program 100 of the present invention. The windows are taken are exemplary and can be created using any visual editor software program, including, for example, Visual Cafe. FIGS. 15-21 show exemplary windows for what the author sees on the authoring side (i.e., configuration editor 110 side), while FIGS. 22-27 show exemplary windows for what the user sees on the delivery side (i.e., the user interface 130 and tutor maker 120 side). As indicated in FIGS. 22-24, the windows can be generated on either a Netscape or Microsoft browser, or alternatively, without a browser for a non-internet application of the present invention.

FIG. 15 shows a configuration editor window 114 that could be used to initiate configuration editor 110 of FIG. 3. The configuration editor window 114 allows an author 32 enter a configuration file location ("D:\TutorMaker\tutorials\test.cfg") into the config. file path field. The configuration editor window 114 also allows the user to access the concept editor 16, the problem editor 16 and the pedagogy editor 20 using the concept editor button 116, the problem editor button 118 and the pedagogy button 122, respectively, in order to create and/or edit the concept, problem, and pedagogy knowledge bases. FIG. 16 shows the concept editor window 124 that can come up on the author's network computer 240 upon clicking the concept editor button 116 of FIG. 15. Concept editor window 124 can list all of the concepts (i.e., concept 1 through concept 6) contained in the configuration 10 chosen in FIG. 15. As shown, concept 3 is highlighted using the cursor and the right hand portion of the window lists a summary of the concept information for concept 3. Concept 3 has a name (concept3), a title (Concept Three), a taxonomy (concept 3 depends on concept 2 and supports concepts 4 and 6), a summary file summarizing concept 3 (sumthree.html) and a detailed file (three.html) providing all of the information the author input about concept 3. The content of concept 3 can be modified by clicking on the "Edit" button 126 or deleted entirely by clicking on the "Delete" button 132.

A new concept can be added to the configuration 10 by clicking on the New button 128 and creating the concept using the OneConceptDialog window 136 of FIG. 17. Similarly, clicking on the show hierarchy button 134 will result in bringing up the ConceptHierarchy window 138 of FIG. 18. If the author is creating a new concept, the OneConceptDialog window 136 will look as it does in FIG. 17 (i.e., without any information in the fields). If the author

18

is editing an existing concept, the existing information for this concept would show up in the fields. As shown in FIG. 17, the author can give the new concept an identifier in the name field 142 and full name in the title field 144. The OneConceptDialog window 136 then allows the author to input the location of a summary HTML file in field 146 and/or the location of a detailed HTML file in field 148. Taxonomy fields 152 provide the important function of allowing the author to indicate the relationship between the new concept being added and existing concepts in the configuration 10. For example, if new concept 7 depends on concept 6, that can be entered into taxonomy field 152. The ConceptHierarchy window 138 of FIG. 18 shows a hierarchy tree 142 of the relationship (or taxonomy) between the concepts in configuration 10. For example, the hierarchy tree 142 shows that concept 3 depends on concept 2 and supports concept 6. As shown, concept 3 supports both concept 6 and concept 4.

FIG. 19 shows an exemplary problem knowledge base window 144 that is brought up by the dynamic knowledge delivery program 100 when the author clicks on the problem editor button 118 of FIG. 15. An author can use problem knowledge base window 144 to create and/or edit problems for the selected configuration 10 of FIG. 15. As shown in FIG. 15, HTML file field 146 shows the HTML file that contains all of the problems, examples and questions related to the chosen configuration 10. Field 148 allows an author to select any concept within configuration 10 from a pull-down list of all the concepts in the associated concept knowledge base. The program 100 can then list all of the problems, examples and questions that the author entered into problem knowledge base for that concept in field 154. Highlighting a particular problem, for example question 41 as indicated in FIG. 19, causes the program 100 to display the information related to that problem on the lower, right side of the problem knowledge base window 144.

FIG. 20 shows the PedagogyEditorDialog window 156 that is generated when an author clicks on the Pedagogy Editor button 122 of FIG. 15. The pedagogy name field 158, indicates the name of the pedagogy for the configuration 10. The concepts field 162 lists all of the concepts contained within the concept knowledge base for the configuration 10 of FIG. 15. FIG. 20 illustrates a way to manually add concepts to a pedagogy and to show the hierarchy using buttons 164 and 166, respectively. When adding a concept using button 164, the add view options box 168 allows the author to add the concept in either detailed or summary form. By adding or deleting concepts, the author can manually generate or edit the pedagogy of pedagogy box 160.

The author can also automatically generate a pedagogy (that will be displayed in pedagogy box 160) by clicking on the auto generation button 150, which can pull up the Auto Pedagogy window 170 of FIG. 21. In an alternative embodiment, the pedagogy can be automatically generated as the author inputs concepts into the configuration. The Auto Pedagogy window 170 will show a list of all the concepts in configuration 10 of FIG. 15 in field 172. The author can then choose the goal concept (i.e., the concept to be taught or conveyed) and which concepts are prerequisites (meaning once that concept has been explained, no further concepts that support a prerequisite concept need be explained). As shown in FIG. 21, the author has chosen concept 1 as the goal concept and concepts 5 and 6 as prerequisite concepts, and the program 100 has placed them in goal field 174 and prerequisite field 176, respectively. In computer science, a "graph" is a general term for any collection of linked objects. The concept knowledge base 11

is a graph built based on the taxonomy of the concepts. The strategy field 178 allows the author to designate the manner in which the graph is traveled between the chosen goal and prerequisites (shown as a "bottom-up" strategy). When the author clicks on the generate pedagogy button 180, the pedagogy generator 50 will automatically generate a pedagogy based on the strategy indicated. In FIG. 21, the strategy, goal and prerequisite indicators tell the pedagogy generator program 50 to start with the goal, end with the prerequisites or an end concept (no supporting concepts), and pick up every concept in between the two.

FIG. 22 shows an exemplary Tutor Maker Login window 182 that includes a username field and a password field to allow the user to log in to the tutor maker program 120. After a successful log in, the tutor maker 120 will generate a Configuration Selection window 184 (shown in FIG. 23) that allows the user to select a configuration 10 by typing the configuration name in field 188 (alternatively, a pull-down menu of all configuration names can be accessed through pull-down button 188). After user 34 has selected a configuration 12 by clicking on the continue button 190, the Presentation Style window 192 of FIG. 24 is generated for the user. The presentation style window 190 includes a list of option circles 192 that allow the user to select the presentation format for the configuration 10. As shown in FIG. 24, the user 34 has selected a standard tutorial format for presentation of the configuration 10. The present invention allows the generation of different presentation formats by its novel concept of completely separating the pedagogy from the content.

Upon clicking the continue button of FIG. 24, the tutor maker 120 will generate a Tutorial Main Frame window 194 as shown in FIG. 25 if a "Standard Tutorial" is selected or an Index Selection Page window 206 as shown in FIG. 26 if a "Surf/Search" selection is made. FIG. 25 is an example of a configuration 10 presented in standard tutorial format with a series of user control buttons on a user control button bar 196 that allow the user to further control the manner of presentation. The content box 198 is where the actual content created by the author is delivered to the user. The table of contents box 202 shows a listing of the concepts which corresponds to the pedagogy. The underlined concepts within the table of contents box 202 indicate hypertext links to the content associated with the underlined concept. The indicator triangle 204 shows the user where within the pedagogy the displayed content of window 198 is located. The control buttons on the user control button bar 196 allow the user to navigate the pedagogy and bring up different content pages. The control buttons can be automatically enabled and disabled as the user navigates the pedagogy. For example, sometimes the "example" button is turned off because no example exists for the concept being displayed in concept box 198. In alternative embodiments, the user control button bar 196 will contain different control buttons for presentation styles.

FIG. 26 is an example of a configuration 10 presented in a surf/search format which can be derived either by selecting "Surf/Search" while at the Presentation Style window 190 of FIG. 24 or by clicking on the "view" button of the Tutorial Main Frame window 194 of FIG. 25. FIG. 26 shows an index of all of the concepts for the configuration 10 available to the user in alphabetical order. The underlining indicates the concepts have hypertext links to the content associated with the concept. FIG. 27 shows a Surf Page window 208 that is produced when the user selects "Asset Allocation" from a surf/search index contained on an Index Selection window 206 as shown in FIG. 26. The user can now see the

content associated with the concept asset allocation in the content box 198 and can link to concepts that are supported by the concept asset allocation in supported concept box 212 and/or concepts that support the concept asset allocation in sub concept box 214. Upon clicking on any of these supported or supporting concepts, the content associated with that topic will pop up in a similar Surf Page window 208.

FIGS. 28-52 are flowcharts illustrating the operation of the dynamic knowledge delivery program 100 of the present invention. FIG. 28 is an overview flowchart of the dynamic knowledge delivery program 100. At step 216, the operation is begun with the author accessing the configuration editor 110 to create one or more configurations 10. The author stores the configurations 10 on the server computer 230 at step 218. The user initiates the tutor maker program 120 on server computer 230 using browser 28 on client computer 220 and accesses tutor maker 120 through network 210. At step 226, tutor maker 120 is executed by server computer 230 to download browser extension 30 to user's client computer 220. The user now has access to the tutor maker 120 and can now query tutor maker 120 as shown by the loop of steps 228 to 242. After making a query, the tutor maker 120 determines whether a pedagogy exists for the query. If not, the tutor maker 120 creates a pedagogy using pedagogy generator program 50 at step 236. The tutor maker 120 then reads the pedagogy and generates HTML for the pedagogy based on the user query and returns the generated HTML to the user over the network at step 242. This loop repeats itself until the user query is an exit query at step 232, whereupon the process is complete.

FIG. 29 is a flowchart of the operation of the configuration editor program 110 of FIG. 4. The configuration editor program 110 allows the author to select one of the concept editor 16, the problem editor 18 or the pedagogy editor 20 at step 244. Depending on which editor is selected, the configuration editor 110 calls the appropriate editor as shown in step 246. FIG. 30 shows the operation of the concept editor 16. Initially at step 248, a menu selection is made by the author. The menu selections of FIG. 30 are delete concept, edit/add concept, save, open, new, and exit. The concept editor includes the new, open, and save functions to allow an author to create, load and save a concept knowledge base, respectively. Once a concept knowledge base has been opened or created (or saved) at steps 252, 254 or 256, the concept editor sends the author back to the menu selection where that author can command the concept editor 16 to add/edit a concept at step 258 (more fully described in FIG. 31) or delete a concept at step 260. To delete a concept, the concept editor 16 will check to see if the concept has supported links at step 262 and if it does, will return a "not valid" message to the author to prevent the deletion of supported concepts. For example, looking at the hierarchy of FIG. 18, the concept editor 16 would send a "not valid" message in response to a request to delete concept 4 because concept 4 is supported by concepts 3, 2 and 1. To delete concept 4, concepts 3, 2 and 1 must first be deleted, or the dependencies upon those concepts must be removed. In the alternative embodiment, the concept editor 16 could simply ask the author whether the author desires to delete concept 3 and all supporting concepts to which the author could answer "yes" and concepts 4, 3, 2 and 1 would be deleted or concept 4 would be deleted and the dependencies removed. If the concept to be deleted does not have supported links, the concept editor 16 will delete the concept from the concept knowledge base (step 264) and will remove all links to this deleted concept (step 266). The concept editor will continue to perform the delete, add/edit, save, open or new functions until the author selects the exit function at step 268.

It should be noted that the interface to the configuration editor 110 described in FIG. 30 may be implemented in any of a variety of ways which support a menu selection mechanism. This includes using the same browser extension techniques described for the tutor maker 120 whereby any standard browser may be extended to implement the interface. More specifically, the techniques outlined herein for the browser extension program 30 may be readily applied on the network computer 240 of FIG. 1 to implement the interface for the configuration editor 110.

In addition, the interface to the configuration editor 110 may be extended to allow for the operation of multiple authors simultaneously developing the same configuration 10. This may be achieved by a simple modification to FIG. 30 wherein every element of the configuration 10 (i.e., all structures described in FIG. 8, FIG. 9 and FIG. 10) are given an additional boolean "in-use" flag which is set to "true" any time that element of the configuration is being modified. With this additional boolean value, step 248 may be modified to first check the indicated element of the configuration 10 that is to be modified to see if its "in-use" flag is set. If so, the menu selection is disallowed. If not, the menu selection is allowed and the in-use flag is immediately set to "true." Once the author completes the operation and returns back to step 248, the in-use flag of the configuration element which was modified in the previous cycle is set to "false" allowing other authors the opportunity to modify that element of the configuration 10.

FIG. 31 illustrates the operation of the concept editor 16 when adding or editing a concept. At step 268, the concept editor retrieves the concept name, the concept title, the dependencies to the concept and HTML references for the concept from the dialog box 136 of FIG. 17. At step 270, the concept editor 16 determines if the concept name exists in the concept knowledge base. If so, the concept editor 16 calls the modify concept function of FIG. 33; if not, the concept editor 16 calls the create concept function shown in FIG. 32. Both of these operations update the concept instances and then store them back in the concept knowledge base at step 276. FIG. 32 shows the operation of creating a concept, including assigning the concept a name, title, and an HTML location using data from the dialog box of FIG. 17. The concept editor 16 then determines if there are any dependent concepts at step 280 (from the information in the dialog box 136) and, if not, goes to step 276 of FIG. 31 and adds the concept to the concept knowledge base. If there are dependencies for the new concept, the concept editor 16 first determines if the dependent concept already exists within the concept knowledge base at step 282. If not, the concept editor calls the create concept function at step 272. If the dependent concept exists in the concept knowledge base, concept editor 16 adds the dependent concept to the dependent links of the concept being created and adds the concept being created to any supported links of the dependent concept at steps 284 and 286. After step 286, the concept editor 16 returns to step 280 to determine if any more dependencies exist until all dependent concepts are in the concept knowledge base and all dependencies are linked.

FIG. 33 shows the operation of editing an existing concept where the initial steps 278 through 286 are the same as creating a concept shown in FIG. 32. At step 288, the concept editor 16 takes the additional action of determining if the modification of the concept resulted in the deletion of some dependencies. If so, at step 290, the concept editor removes any dependent concepts from the concept dependent links and removes concepts from any dependent concept supported links. For example, if ten dependencies

existed before the modification of the concept and all ten were deleted during the modification of the concept, then the set of old pointers which linked the modified concept to the ten dependent concepts would be removed, and each of the ten dependent concepts would have its supporting link back to the modified concept removed.

FIG. 34 shows the operation of the problem editor 18 that the user initiates by clicking on the problem editor button 118 of FIG. 15. Again, at step 292, the user makes a menu selection that causes the problem editor 18 to perform a function on the menu, shown as delete, edit, save, create or load a problem knowledge base, until the selection is "exit" at step 294, upon which the problem editor 18 closes. At step 296, the problem editor 18 can load problem knowledge base by parsing the problem knowledge base using Backus-Naur Format (BNF) and calling the update problem function (FIG. 35) to save each parsed problem. In the preferred embodiment, a problem knowledge base is a single HTML file (rather than separate HTML files for every single example, problem and question). In order to be able to later separate the individual problems, questions and examples from one another, the ones related to a particular concept must be parsed out from the problem knowledge base HTML file. In one embodiment, the BNF definition (i.e., syntax, grammar and semantics) for parsing the problem knowledge bases is as follows:

BNF for the Syntax for Problem Delimiters

"()"	indicates a grouping of items. If ' ' is present, then exactly one of the items is to be selected.
"{ }"	indicates a list. Repeat pattern with { } 0 or more times if "*" is used; repeat 1 or more times if "+" is used.
"[]"	indicates an optional item
"..."	indicates a comment

uppercase is for non terminals lowercase is for terminals (also, items enclosed in single quotes are terminals).

EXTRA-SYNTACTIC RULES:

1. TUTORFILE commands can appear within any general HTML file.
 2. Attributes of an HTML element (including DIV) can appear in any order. This applies to QUES-HEAD, EXAM-HEAD, EXPL-HEAD and CHOICE-HEAD.
 3. White space must SEPARATE attribute/value pairs in an HTML element (see note 2 above). There can also be optional white space before the final '>' of an HTML element.
 4. White space is optional between elements in HTML. This applies to productions QUESTION, EXAMPLE, EXPLANATION and CHOICE below.
 5. All attribute/value pairs, e.g. id="t1", can have WS around the "=", and can enclose the value in either " " or ' ' or use no quotes at all (if no quotes are used, then no WS may occur around the "="). Below, we show all attribute values enclosed in " ". This applies to the ID, CONCEPT, TYPE and CONTEXT productions below.
- TUTORFILE ::= { COMMAND } * -- optional white space between commands
- COMMAND ::= (QUESTION | EXAMPLE)
- QUESTION ::= QUES-HEAD HTML ::= [EXPLANATION] {CHOICE}+ DIV-END
- EXAMPLE ::= EXAM-HEAD HTML [EXPLANATION] DIV-END
- EXPLANATION ::= EXPL-HEAD HTML DIV-END
- CHOICE ::= CHOICE-HEAD HTML DIV-END [EXPLANATION]

23

```

QUES-HEAD ::= '<DIV' 'class="question"' ID TYPE
CONCEPT '>'
EXAM-HEAD ::= '<DIV' 'class="example"' ID CONCEPT
'>'
EXPL-HEAD ::= '<DIV' 'class="explanation"'
[CONTEXT] ['override'] '>'
CHOICE-HEAD ::= '<DIV' 'class="choice"' ['correct'] '>' --
note that only CHOICE-HEAD for a group of CHOICES
-- for a given QUESTION may be marked 'correct'.
ID ::= 'ID=" ' IDENTIFIER ' "'
CONCEPT ::= 'concept=" ' IDENTIFIER ' "'
TYPE ::= 'type="mc"' -- eventually, more types
CONTEXT ::= 'context=" ' IDENTIFIER {WS IDENTIFI-
FIER} * ' "' -- note there MUST be white space between
identifiers -- here (i.e., they are white space separated).
DIV-END ::= '</DIV>'
IDENTIFIER -- alpha numeric, also may have '-' and '_' --
but NO white space

```

HTML -- any HTML text, including no text (empty)

As the problem knowledge base is parsed, each problem is translated into a corresponding problem data structure 72 (see FIG. 9). The update problem function of FIG. 35 shows that the problem editor 18 reads a problem to determine if it is a new problem at step 322, and if so retrieves the type of problem (i.e., multiple choice question, etc.) and builds the type class for the problem at step 326 with the name, concept reference, HTML reference and explanation passed to it as parameters. If the problem is not new, then the problem editor 18 retrieves the existing problem, removes it and replaces it with the new updated problem at step 324. At the last step 328, the problem editor 18 ensures every problem is associated with a particular concept by searching the concept knowledge base to fill in any links to concepts that are missing. This provides the links between the problems and the concepts to which they relate.

With reference to FIG. 34 again, the new selection 300 and save selection 304 cause the problem editor 18 to create an empty problem knowledge base (step 302) or to save a problem knowledge base (step 306), respectively. If the user selects edit at step 308, the problem editor 18 allows the user to load a problem HTML file into an HTML editor at step 310 to edit the problem. The problem knowledge base is then updated by parsing the output of the HTML editor to place the problem in the proper format and by calling the update problem function as previously described.

FIG. 36 shows the operation of the pedagogy editor 20 that a user can initiate by clicking on the pedagogy editor button 122 in the configuration editor 114 of FIG. 15. That operation can bring up the pedagogy Editor Dialog window 156 of FIG. 20 to allow an author to create or modify a pedagogy (manually or automatically). The Pedagogy Editor Dialog window 156 can include a pull-down menu or individual buttons that allow a user to activate the functions of FIG. 36. In the embodiment shown in FIG. 36, the menu selection includes save, load, new, auto generate, edit/add and delete functions. As shown, the performance of any of the steps returns the author to the menu selection 330 to perform any of the other functions and, after any combination of these, the author can exit at step 332. Steps 334, 336, and 338 allow an author to load an existing pedagogy from a storage location, create a new pedagogy by linking concepts in any chosen manner, and to save a pedagogy for a particular configuration, respectively. At step 340 the dynamic knowledge delivery software 10 can automatically generate a pedagogy. The initial step 342 can include the option of allowing the user to select a generation type. The generation type is the method of presenting the concepts

24

based on the taxonomy. For example, the generation type might be a bottom-up or top down or some other relational pattern. The user can also optionally select the generation parameters at step 344 which will set the goal and the prerequisites. The dynamic knowledge delivery program 100 will then call the autogeneration routine that will arrange the concepts based on the generation type and the goal (start) and prerequisites (end) at step 350, as shown in FIG. 37. Finally, the resulting pedagogy is added at step 352. The edit/add step 354 calls the edit/add function 360 of FIG. 39, while the delete function at step 356 calls the delete function 370 of FIG. 38.

FIG. 37A shows the operation of the Pedagogy Generator function that occurs upon the author's selection of the auto generator button 150 of FIG. 20. As shown in FIG. 37A, at steps 362 and 364 pedagogy generator 50 sets the goal concepts and prerequisites that were selected or passed in (e.g., as described in FIG. 36). The goal concepts and prerequisite concepts must be chosen (either manually by the author or automatically by the dynamic knowledge delivery software) to allow the generation of the pedagogy. At step 366 the pedagogy generator 50 determines whether the context concepts were specified and, if so, adds the context to the prerequisites concepts at step 368. This is an optional step where, if the user or author has specified particular concepts types only to be delivered, the pedagogy generator 50 will only search on those particular concepts types within the concept graph and only those particular concept types will be included in the resulting subgraph of concepts to be delivered to the user. If the author is creating the pedagogy, then the author chooses the type. If the TutorMaker is generating a pedagogy for a response to the user, then it can be based on the user's type (i.e., if user technical vs. salesman etc.) This essentially operates as another type of prerequisite. The concepts have been labeled with these "type" labels in the concept knowledge base. At step 370 the pedagogy generator 50 then collects the subgraph of concepts according to a searching algorithm incorporated into the pedagogy generator 50. The collection of the related concepts as a subgraph can be done by recursive descent (e.g., depth first, breadth first, stopping at concepts that have no children, stopping at prerequisite concepts or prerequisites based on user's background, etc.). At step 372, the concepts within the collected subgraph are ordered based on the generation type selected by the user or programmed into the pedagogy generator 50. Optionally, the searching and traversing of the graph can also be guided by parameters included from a user model that is accessed prior to searching the graph.

As noted, the pedagogy generator 50 includes a graph searching algorithm that navigates, according to the type of algorithm, the graph of concepts that is based on the taxonomy. Any graph searching algorithm can be used in the present invention. A "graph" is a standard computing term for a series of linked nodes, that are typically acyclic for the present invention (i.e., the graph does not wrap back around on itself), where each node represents a concept. The links are connected to one another according to the relationship of each concept to every other concept (i.e., the taxonomy).

Graphs are typically represented so that upper nodes are parents, lower linked nodes are children of the parent nodes, and nodes on the same level are sibling nodes. The pedagogy generator 50 sets the goal concept node as a starting node and the prerequisite concept nodes as ending points and uses the graph searching algorithm to search the graph using its designated method. For example, one method is a depth first search that requires the search to go from parent to its child

(if multiple children, to a particular child based on its location in the graph), to its child, until no more children are reached, and then back to the upper-most node that has other children not searched, and repeats until all nodes have been searched. Another searching method is a breadth first search which requires the search to start at a parent node, search all siblings of that parent node first, and then go to the first parent node and search all of its children. Basically, the pedagogy generator uses a starting point (the goal concept), ending points (the prerequisite concepts), and a generation type to define what search method to use to create a pedagogy by traversing through the graph of concepts and ordering the presentation of concepts based on the order each node (or concept) is encountered during the search. The graph was built in the concept editor based on the relationships defined between the concepts as the concepts were entered.

Essentially, the pedagogy generator 50 traverses the graph of concepts for a particular configuration and collects a subgraph of the concepts based on the goal, prerequisite and generation type parameters. This subgraph is the pedagogy that defines what concepts will be presented and in what order the defined concepts will be presented to the user. Thus, the pedagogy generator 50 will automatically select both the relevant concepts to display and the appropriate order in which to display them based on the dependencies and relationships of the associated graph.

To further illustrate the sorting performed during automatic pedagogy generation function described in FIG. 37A, FIG. 37B shows an exemplary configuration delivery 374 having a pedagogy that was automatically generated based on the taxonomy of FIG. 37C (for the six concepts in the example configuration 376). FIG. 37C shows that the author assigned the hierarchy or taxonomy 378 among the six concepts in the configuration 376 (using the configuration editor) as follows: concept 6 depends on concepts 3 and 4; concept 3 depends on concept 2; concept 4 depends on concepts 3 and 1; and concept 6 depends on concepts 3, 4 and 5. FIG. 37B shows the pedagogy automatically created based on the FIG. 37C taxonomy based on a goal concept of concept 6 and no prerequisite concepts (i.e., include every concept in the configuration). The steps of FIG. 37A, the pedagogy generator 50 will generate the pedagogy that is presented in FIG. 37B. The pedagogy of FIG. 37B shows that in order to explain/understand concept 6, concepts 3, 4 and 5 must be explained in that order. In order to explain concept 3, concept 2 must be explained. In order to explain concept 4, concept one must be explained, and then concept 5 is explained. At this point, all of the prerequisites to concept 6 have been explained and concept 6 can be delivered. As shown in FIG. 37B, concept 4 can include again concept 3 with its supporting concept 2 in summary form. FIG. 37B illustrates a "bottom up" generation type of delivery format (e.g., prior to the explanation of a higher level concept such as a computer, the user must first have delivered an explanation of a set of lower level concepts such as transistors and software programs).

FIG. 38 shows the operation of the delete element in the pedagogy function that allows an author to delete a subsection within a tutorial delivery. For example, if the author wants to delete concept 2 from the Unit Explaining concept 3, the author can delete concept 3 from that unit, or even delete the unit entirely. In step 382, the configuration editor 110 can first query whether the pedagogy element to be modified is a "unit" (i.e., a set of concepts versus a single concept). If not, the program allows the author to delete the pedagogical element (i.e., concept) at step 384. If so, at step

386, the configuration editor 110 can query whether the entire unit should be deleted or just some subset of concepts within the unit. If the entire unit is to be deleted, the configuration editor 110 can further query whether the children concepts of the concepts to be deleted are also to be deleted and can delete them again at step 384. This involves searching the pedagogy and grabbing all of the supported concepts (or children). If the entire unit is not to be deleted, then the parent concept of all the children concepts are set to the unit's parent concept at step 390 prior to deletion at step 384. This effectively adds the children concept to the unit which enclosed the deleted unit.

FIG. 39 shows the add/update pedagogy function 380 that allows concepts to be added to a pedagogy by the author. The configuration editor 110 first queries whether the pedagogical element is a new element (i.e., to be added) at step 392 to determine whether the element already exists in the pedagogy. If so, then the parent of the pedagogy element is set as shown in FIG. 10 at step 394. The configuration editor 110 then queries whether the new element is a concept at step 396. This is included because if a concept is added, it should be linked to its parent concepts if applicable. If the element is a concept, then a link and view type is set for that concept at step 400 and the function is complete. The configuration editor 110 then queries whether the new element is a problem concept at step 400, and if so, sets the problem size and view type at step 404. If the new element is not a concept or problem, then it is a unit and the unit name and optionally the HTML reference are set at step 406.

FIGS. 40 through 50 relate to the tutor maker 120 and the user interface 130. FIG. 40 shows an exemplary flow diagram of the operation of the tutor maker. Upon start-up, the tutor maker 120 first runs an initialization phase to initialize various parameters that can include the configuration default directory, session timeout, an empty session database and an empty tutorial database at step 410. In one embodiment, the tutor maker 120 can track where all the configurations 10 using a default directory that will contain all of the accessed configurations 10. The timeout feature simply defines how long user's can access a configuration without initiating an action before the session will be automatically terminated by the tutor maker 120. Further, each time a user logs into the tutor maker 120, the user gets a user session. The tutor maker 120 then initiates a low priority thread at step 412. The thread is an independent program that will automatically remove stale user sessions. The low priority option means that the thread will only operate when all users have been currently served with all their current outstanding requests. Finally, at step 414, the tutor maker 120 calls the action handler 42, which then waits for an action to come in from the user over the internet or network.

FIG. 41 shows a flow diagram outlining the action handler 42 functionality for one embodiment. At step 416, action handler 42 waits for an action to come in from the user over the internet or network and gets the information from the request at step 418. Each request is paired with a unique identifier (called a "cookie" for the internet). Part A of FIG. 41 indicates the action handler 42 synchronizing on a user session database. If the action handler 42 is able to locate a user session database associated with the received cookie at step 420, the action handler moves to part B (which describes the synchronization on a particular user session). The synchronization is a standard computer science techniques used in concurrent programming to prevent deadlock problems that arises when two or more users attempt to simultaneously access the same data. In part A, the tutor

maker 120 provides for synchronizing on the user session database to lock that user session database prior to determining if the cookie is associated with a particular user session. Thus, if the user session database is associated with a session in the cookie at step 420, then go to part B and synchronize with the session at step 430. If not, then the tutor maker 120 will check to see that the user is a valid user and authenticates the user at step 422. If the user is authentic, then the action handler 42 creates a new user session with a unique ID, a new user object with the session (as shown in FIG. 13) at step 424. This creates the user session and inserts it into the user session database. The configuration is then selected at step 426 and the action handler 42 sends the session ID back to the user over the network at step 428 (e.g., as a cookie for an internet application). The action handler 42 is now once again in a ready state to receive a request.

For requests that are for an existing user session, prior to performing the action, the action handler must synchronize on the existing user session at step 430. The action handler 42 will return actions at step 428 until the user requests a logout and at step 434 the action handler 42 logs off by deleting the shared configuration 10 at step 436, optionally saving the history of actions for the user session at step 438, marking the session as invalid for removal at step 440 (this simply marks it to be cleaned up by the low priority task that was previously initiated), and generating a message HTML indicating the session has been terminated at step 442.

For an established session, a user either selects a configuration at step 444, relocates from one configuration to another at step 446, or takes a tutorial action in a configuration at step 448. If the user selects a configuration (i.e., the user is logged in but has not yet selected a configuration), the action handler 42 retrieves the configuration 10 (step 450), determines whether a user ID and configuration pair exists (step 452), and if so, generates a relocation HTML because the user was already viewing the configuration from another location. This HTML asks the user if they want to relocate the user session. In this manner the configuration in use by the user at the different location can be terminated and restored at the new location. The dynamic knowledge delivery program 100 can also include the capability to lock out a user on the tutor maker side at server computer 230 from accessing a particular configuration 10 if that configuration 10 is in use by an author on the configuration editor side at network computer 240. The dynamic knowledge delivery program 100 can recognize automatically that an update is being performed by the author to the configuration on the configuration editor side changed and, therefore, locks out the internet or network side so that a user cannot access a configuration being updated. This locking is achieved by creating a temporary file called a lock file in the directory where the configuration resides. Before allowing access to a configuration, the directory is checked to see if a lock file is present. If not, access is granted and a lock file is created. Once the configuration file is no longer needed the lock file is deleted.

If the user ID/configuration pair does not exist, the load configuration function 480 is called at step 456 and the configuration is loaded as described more fully in FIG. 42. The action handler 42 then generates the main display HTML associated with the configuration chosen at step 460.

If the user is simultaneously logged on, the user answers the relocation prompt (see step 460) at step 446. If the user does not want to relocate the user session, the action handler 42 responds to the user with the configuration selection HTML at step 462. If the user does want to relocate, the

action handler 42 calls the delete configuration function 500 at step 464 to delete the configuration that is active at the other location (more fully described in FIG. 44). The action handler 42 then performs very similar functions as when the user logs-out of saving the history at step 466, mark the session for removal at step 468, optionally save the user's history at that session, and creates a new user session with a unique ID and user object session (step 470). In other words, the action handler 42 deletes the old session, makes a new one, and loads it by calling the load configuration function 480 (shown in FIG. 42). The action handler then generates the main display HTML for the loaded configuration. The loaded configuration will include all of the user history from the old session so that the user is looking at the configuration at the same state the user left it at the other location.

For any other tutorial action (e.g., moving through a configuration) at step 448, the action handler 42 receives the requested action, retrieves the user's session at step 476 and then retrieves the pedagogy example, or test question, or calls the generate pedagogy function, etc. and then generates the HTML and returns it as a response at step 478.

FIG. 42 illustrates the load configuration function 480 that first queries whether the configuration has previously been loaded by an user at step 482. If it has, then the tutor maker 120 retrieves the shared configuration object from the tutorial database 46 at step 484 and increases the shared configuration (SC) increment count by one at step 486. If the configuration is not in use, the tutor maker 120 creates a new shared configuration object at step 486 and adds it to the shared configuration database at step 488. At this point the user session is synchronized on the tutorial database. The user session is then synchronized on the shared tutorial object by setting the SC count to one at step 490 and then calling the load concept knowledge base (step 492), the load problem knowledge base (step 494) or load pedagogy knowledge base (step 496) as more fully described in FIG. 43.

FIG. 43 shows the load knowledge base function 520 performed by the tutor maker 120 that initially queries whether the particular knowledge base (i.e., concept knowledge base, problem knowledge base, pedagogy knowledge base) has been previously loaded at step 498 to synchronize on the tutorial database. FIG. 43 represents the method used to load all three knowledge bases associated with a configuration. If so, the tutor maker 120 retries the shared tutorial object corresponding to the requested knowledge base at step 502, increments the count for that knowledge base at step 504, and the shared tutorial object is set in the appropriate slot of the shared configuration 97 at step 506. More specifically, if the concept knowledge base is being loaded, then the shared configuration object 97 will have its concept knowledge base reference set to point to the shared tutorial object created for the concept knowledge base. In this way, each user session refers to a shared configuration structure, which in turn refers to shared tutorial objects, each of which keeps a count of the current number of users currently using each component of a configuration. Thus, when the count of a shared tutorial object (decremented as each user logs off) reaches zero, that object may be deleted. If the knowledge base has not been previously loaded, it is then loaded as a shared tutorial object at step 508, which is then added to the corresponding links within the tutorial database at step 510. For each added shared tutorial object, the increment count is set to one at step 512 and then set in the appropriate slot of the shared configuration 97 at step 506. As shown in FIG. 43, part A involves synchronizing on the tutorial database, while part B involves synchronizing on the shared tutorial object.

FIG. 44 describes the delete configuration function 500 used when the last user for a particular configuration logs off in order to delete the configuration from the tutorial database. At step 516, the tutor maker 120 determines the shared decrement count for the configuration in question and checks whether the count for the configuration is equal to zero at step 518. If not, the count is decremented by one. If the SC count is zero after the removal of this instance of the configuration, then the parallel processing for each of the concept knowledge base, problem knowledge base and pedagogy knowledge base occurs. For example, a concept knowledge base might be used on other loaded configurations so that while the configuration (i.e., the particular set of concept, problem and pedagogy knowledge bases) count may now be zero, the particular concept knowledge base used for the configuration might not have a count that goes to zero (e.g., if it is shared between at least one other configuration). Thus, for each of the concept knowledge base, the problem knowledge base, and the pedagogy knowledge base, the delete knowledge base function 530 (shown in FIG. 45) performs the similar decrement analysis as done

at step 552. The tutor maker 120 can then go get the HTML from either a configuration file (step 554) or generated based on a pedagogy generation (step 556). If the HTML will be extracted from a configuration file, the tutor maker 120 accesses the user object from the user session at step 558, accesses the configuration from the user object at step 560, accesses the problem or concept from the configuration 10 and extracts the HTML from the concept at step 562, and outputs the HTML at step 564. If HTML is not to be extracted from a configuration, then the tutor maker may have to generate a pedagogy (step 556). If generating a pedagogy, then the tutor maker 120 can first access a user object from the user session at step 568, access the configuration from the user object at step 570, extract the concept(s) and prerequisite(s) from the configuration (and optionally the user history information) at step 572, and call the pedagogy generator (see FIG. 37A). The resulting pedagogy is rendered using either the HTML associated with elements in the pedagogy or an "on-click" event handler. An on-click event handler can be created as follows:

General HTML Header Command Elements:

```
<!DOCTYPE>
<HEAD>
<TITLE>
<BASE>
<META>
```

Event Handler Format:

```
Click: < HTML-ELEMENT-NAME on Click="event-Function (parameter)" >
Mouse: < " on MouseOver=" " >
      < " on MouseOut=" " >
```

etc.
links HTML ELEMENT TO event-Function in script, and passes arguments to that function

Shared Data Objects:

```
Current HTML (3.2): use <FORM> element with "hidden" <INPUT>
elements. Future HTML versions may use <OBJECT> element.
```

for the overall configuration prior to deleting them (steps 522, 524, and 526). Finally, because the shared configuration count is zero, the shared configuration is marked as invalid at step 528 and cleaned up.

FIG. 45 shows the delete knowledge base function 530 that is applicable for each of the three knowledge bases. Similarly to the steps in FIG. 44, for the knowledge base being deleted the decrement count of the shared tutorial object corresponding to the knowledge base is determined at step 532, and if it is not zero, the count is decremented by one and if it is zero, the knowledge base is marked as invalid at step 536.

FIG. 46 shows the generation of HTML function 540 to describe the generation or building of HTML generally on the tutor maker 120 side of the operation of the present invention. This can include extracting HTML to generate a page for the user, generating HTML for particular formats, generating features on an HTML page such as toolbars, etc. At step 542, the tutor maker 120 can either output a header or not, and if so, outputs a general header HTML at step 544 that can include a style sheet. A style sheet is mechanism by which any HTML web page that has been generated has certain styles automatically overlaid onto it. By automatically associating a style sheet will all generated HTML, the present invention can control the look and feel of the pages that are presented to the user. After the style sheet is loaded or not at steps 548 and 548, a script call can be made at set 550. A script, for example javascript, is loaded from a link

In one embodiment, the present invention uses the above-referenced general HTML commands (shown in angled brackets) are used in the header. The event handler format is contained within an angle bracket having some HTML element name (e.g., highlight), followed by "on-click" and the text for the function call in the browser extension 30. This allows the overriding of the typical response when a user clicks on this particular HTML element. Thus, the tutor maker 120 overrides by providing the specific on-click command. Similarly, specific events can occur for mouseovers, and other standard HTML events. Thus, some HTML elements allow the overriding of typical responses on certain events (such as, on-click, on mouse-over, and on mouse-out) that are pre-defined for HTML. The shared data objects (e.g., a toolbar) are global variables that can be hidden input elements. One of ordinary skill in the art will appreciate how to create an on-click type event handler. Thus, the results (i.e., the graph) generated can be printed out, for example as hyperlinks. In one embodiment, the hyperlinks are separated into two frames and the normal browser hyperlink is overridden using the above described on-click mechanism. Finally, at step 578, the tutor maker 120 can create a closer and close the open HTML elements at step 580. A closer is a commonly used mechanism to end an HTML generation event. As shown in FIG. 46, tutor maker 120 uses style sheets and scripts to extend the browser in an internet application of the present invention.

FIG. 47 describes the operation of the toolbar HTML generation function 582. In one embodiment, the present

invention can include a toolbar in each HTML page generated for a user to simplify use of the present invention. The toolbar can be generated by outputting a header with a script and style sheet at step 584. At step 586, the tutor maker 120 can include a global form onto each HTML page so that the form is shared among all generated frames. In another embodiment, the global data of a toolbar can be accomplished using the <object>-element of HTML (proposed for version 4.0 of HTML). At step 588, the tutor maker 120 outputs the shared data objects for the state information on the client computer 220 side (i.e., the browser side for an internet application) and finally, outputs the button image that is linked to an event handler at step 590.

FIG. 48 shows the operation of the event handler that is part of the user interface 130. The event handler waits for an event at step 594 and upon receipt of the event calls that type of event. FIG. 49 shows eight exemplary events that can be incorporated into the present invention. Step 596 shows a call tutor maker event that can be accomplished using servlet technology from Sun Microsystems. The event handler accesses the required shared data in servlet form at step 598 and submits the form to invoke the servlet over the internet at step 600. It should be understood that a variety of methods exist to submit data over a network, and the servlet form method described here is simply one method for use with the internet. The method described here will not cache and therefore it is important to use a form-submit or similar methodology for a browser-specific implementation of the present invention over the internet. Other actions include an exit function shown include an exit servlet at step 602, an update hierarchy at step 603, a concept link at step 604, a button state at step 605, a pop-up window at step 606, a change display mode at step 607 and an exit at step 608 to exit the event handler by calling a pop-up handler to kill all pop-up windows at step 609 and sending a call servlet event with a 'logout' action at step 610. The update hierarchy event is accomplished by modifying the old image to remove the present location indicator at step 611 and modifying the new image to include the present location indicator at step 612. The concept link involves extracting the servlet data from a concept link event handler at step 613 and sending a call servlet event containing that extracted data at step 614. The button state event is initiated by setting the shared data objects to reflect the current button state at step 615 and replacing the button image at step 616. The pop-up window event simply calls a pop-up handler with the servlet data at step 617 (as shown more fully in FIG. 50), while the change display mode similarly calls a mode handler with the servlet data at step 618 (as shown more fully in FIG. 50).

FIG. 49 describes the pop-up handler function 620 including an initial request from the user interface 130 to either terminate the pop-up window (step 624) or to access a pop-up window (step 626). If the pop-up window is requested, the user interface 130 can first check to determine if the pop-up window is already present at step 628, and if so, can re-use the existing pop-up window at step 632. If not, the pop-up window is generated in its default state at step 630. Finally, the pop-up window contents can be updated by calling a servlet via the servlet data form in the toolbar frame at step 634.

FIG. 50 describes the mode changer function 640 that changes the mode of the browser display. At step 642, an event handler can be called to set the new button states for the new mode. The user interface can determine whether the new mode requires terminating any/all pop-up windows at step 644, and if so, can call the pop-up handler of FIG. 49 to do so as shown at step 646. At step 648, the user interface

130 sets the shared data to reflect the new mode information. The call servlet function is then performed at step 650 to update the browser display. Multiple call servlets can be made if the new display has multiple frames. For example, each of the frames can call the event handler with the appropriate servlet data.

Although the present invention has been described in detail, it should be understood that various changes, substitutions and alterations can be made hereto without departing from the spirit and scope of the invention as described by the appended claims.

What is claimed is:

1. A system for delivering information to a user of a computer having a processor and a memory, comprising:

a configuration database stored on the memory and comprising a plurality of concepts and including a hierarchy that defines the relationships among the plurality of concepts;

a knowledge delivery program executable by the processor to:

create a presentation format for the configuration; and deliver at least a portion of the plurality of concepts in the configuration to the user according to the presentation format.

2. The system of claim 1, wherein the knowledge delivery program automatically creates the presentation format for the configuration based on the hierarchy.

3. The system of claim 1, wherein the presentation format is created in response to a user request and based on a set of inputs from the user.

4. The system of claim 3, wherein the knowledge delivery program further comprises:

a configuration editor software program executable to: allow an author to create the concept knowledge base and the problem knowledge base; allow the author to build the hierarchy as the author inputs the concepts into the concept knowledge base; and store the configuration in the memory; and

a tutor maker software program executable to:

allow the user to access the configuration; retrieve concepts and problems within the configuration based on the presentation format; and send the concepts and problems within the presentation format to the user according to the presentation format;

and wherein each configuration further comprises:

a concept knowledge base containing the plurality of concepts;

a problem knowledge base containing a plurality of problems relating to the concepts in the concept knowledge base; and

a presentation format knowledge base containing at least one presentation format for the concepts in the concept knowledge base.

5. The system of claim 4, wherein the set of inputs comprises the user's familiarity with the plurality of concepts, a level of detail the user wants, or a format in which the user wants the at least a portion of the plurality of concepts presented.

6. The system of claim 5, wherein the concepts input by the author include content, wherein the content can comprise descriptive content, referential content, or both, and wherein the hierarchy represents a portion of the referential content of the configuration.

7. The system of claim 4, wherein the knowledge delivery program further comprises a user interface software program that the user can use to access the tutor maker program.

33

8. The system of claim 7, wherein the user interface further comprises:

- a browser that enables browsing of the network; and
- a browser extension, comprising an event handler executable to allow the user to generate an event to be acted upon by the tutor maker program.

9. The system of claim 4, wherein the configuration editor program further comprises:

- a concept editor executable to allow the author to add, delete or modify concepts within the concept knowledge base and to create a hierarchy for the concepts in the concept knowledge base, further comprising a concept reader executable to retrieve existing concepts within the concept knowledge base;
- a problem editor executable to allow the author to add, delete, or modify problems within the problem knowledge base, further comprising a problem reader to retrieve existing problems within the problem knowledge base;
- a presentation format generator executable to automatically generate a presentation format for the configuration; and
- a presentation format editor executable to allow the author to manually create a presentation format or to modify an automatically created presentation format within the presentation format knowledge base, further comprising a presentation format reader executable to retrieve an existing presentation format.

10. The system of claim 9, wherein the concepts within the concept knowledge base are compatible with or convertible to a format compatible with delivery of information from a global computer network to a general purpose browser, and wherein the concept reader is further executable to create a pointer for each concept that will point to all content associated with each concept.

11. The system of claim 4, wherein the tutor maker program further comprises:

- an action handler program executable to receive a request relating to a requested configuration from the user and process the request;
- a tutorial manager program executable to:
 - determine whether the content of the requested configuration has previously been loaded, and if not, load the requested configuration; and
 - determine whether a user session has been established, and if not, establish a user session;
- a tutorial database containing each loaded configuration;
- a user session, executable to:
 - establish a thread for each user to facilitate synchronization among multiple users;
 - process the request by accessing the loaded requested configuration;
 - generate a response; and
- an a format compatible with delivery of information from a global computer network to a general purpose browser generator executable to:
 - receive the response from the user session;
 - place the response in a format compatible with delivery of information from a global computer network to a general purpose browser format; and
 - transmit the a format compatible with delivery of information from a global computer network to a general purpose browser response back to the user.

12. The system of claim 11, wherein the action handler processes the request by forwarding the request to the user session.

34

13. The system of claim 11, wherein the tutorial manager is further executable to track a set of referential content associated with each concept in the concept knowledge base, each problem in the problem knowledge base and each presentation format in the presentation format knowledge base to ensure that only one instance of each knowledge base is loaded at any one time.

14. The system of claim 11, wherein the tutorial manager program further comprises:

- a concept reader executable to retrieve existing concepts within the concept knowledge base;
- a problem reader to retrieve existing problems within the problem knowledge base; and
- a presentation format reader executable to retrieve an existing presentation format.

15. The system of claim 11, wherein the tutorial database is a collection of pointers that point to a collection of a format compatible with delivery of information from a global computer network to a general purpose browser pages that make up the loaded configuration.

16. The system of claim 11, wherein the user session is further executable to build a user model for each user that contains at least a historical data base of the user's interactions with the tutor maker program.

17. The system of claim 11, wherein the action handler processes the request by either forwarding the request to the tutorial manager if a user session has not previously been established, or forwarding the request to the user session if a user session has previously been established.

18. The system of claim 11, wherein the tutor maker further comprises a presentation format generator executable to generate a user-defined presentation format for the loaded configuration.

19. The system of claim 8, wherein the network is the internet and the browser is an internet browser, and further wherein the event handler further comprises:

- a pop-up handler executable to receive an input from the event handler and, if a pop-up up window is associated with the input, creates the pop-up window for viewing by the user;
- a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, creates the mode change for viewing by the user; and
- an action sender executable to send a request based on the input from the event handler to the tutor maker program.

20. The system of claim 1, wherein the concepts are delivered to the user in a tutorial format.

21. The system of claim 1, further comprising a plurality of configurations, wherein any number of the plurality of configurations can be accessed by any number of users simultaneously.

22. The system of claim 1, wherein the knowledge delivery program creates the presentation format automatically by:

- creating a graph representing the concept knowledge base based on the hierarchy input by the author; and
- searching the graph using a graph searching algorithm to place concepts within the presentation format based on the order found by the graph searching algorithm.

23. The system of claim 22, wherein the knowledge delivery program is further executable to allow the author to select a goal concept and at least one prerequisite concept, and wherein the graph searching algorithm uses the goal concept as a starting point and the prerequisite concepts as

35

an ending point and collects a subgraph of the concept knowledge base graph, the subgraph defining the presentation format so that both a set of relevant concepts and an order to display the relevant concepts is generated automatically.

24. A system for delivering information to a user of a computer over a network, comprising:

- a server computer, comprising:
 - a server processor; and
 - a server memory;
- a client computer connected to the server computer via the network, comprising:
 - a client processor; and
 - a client memory;
- a network computer connected to the server computer via a server computer-network computer network, comprising:
 - a network processor; and
 - a network memory;
- a configuration editor program stored on the network computer memory and executable by the network computer processor to allow an author to:
 - create a configuration database comprising a plurality of concepts; and
 - build a configuration hierarchy defining the relationships among the plurality of concepts;
- a presentation format generator program stored on either the server computer memory or the network computer memory and executable to:
 - automatically create a presentation format for the configuration based on the configuration hierarchy; and
- a tutor maker program stored on the server computer memory and executable by the server processor based on requests from the user to:
 - deliver at least a portion of the plurality of concepts in the configuration to the user according to the configuration presentation format.

25. The system of claim 24, further comprising a user interface program stored on the client computer memory and executable by the client processor to allow a user to send requests to and receive responses from the server computer via the network, and wherein a user can send an input that initiates the automatic generation of the presentation format.

26. The system of claim 25, wherein the network is the internet and the user interface program further comprises:

- an internet browser that enables browsing of the internet; and
- a browser extension, comprising an event handler executable to allow the user to generate an event to be acted upon by the tutor maker program.

27. The system of claim 26, wherein the event handler further comprises:

- a pop-up handler executable to receive an input from the event handler and, if a pop-up window is associated with the input, create the pop-up window for viewing by the user;
- a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, create the mode change for viewing by the user; and
- an action sender executable to send a request based on the input from the event handler to the tutor maker program.

28. The system of claim 24, wherein each configuration database further comprises:

36

a concept knowledge base containing the plurality of concepts;

a problem knowledge base containing a plurality of problems relating to the concepts in the concept knowledge base; and

a presentation format knowledge base containing at least one presentation format for the concepts in the concept knowledge base.

29. The system of claim 24, wherein the configuration editor software program is further executable to:

allow an author to create the configuration by inputting the concepts into the concept knowledge base and inputting the problems into the problem knowledge base;

allow the author to build the hierarchy as the author inputs the concepts into the concept knowledge base; and store the configuration in the server memory.

30. The system of claim 24, wherein the concepts input by the author include content, wherein the content can comprise descriptive content, referential content, or both.

31. The system of claim 24, wherein the configuration editor program further comprises:

a concept editor executable to allow the author to add, delete or modify concepts within the concept knowledge base and to create a hierarchy for the concepts in the concept knowledge base, further comprising a concept reader executable to retrieve existing concepts within the concept knowledge base for deletion or modification;

a problem editor executable to allow the author to add, delete, or modify problems within the problem knowledge base, further comprising a problem reader to retrieve existing problems within the problem knowledge base for deletion or modification; and

a presentation format editor executable to allow the author to manually create a presentation format or to modify an automatically created presentation format within the presentation format knowledge base, further comprising a presentation format reader executable to retrieve an existing presentation format for modification.

32. The system of claim 31, wherein the concepts within the concept knowledge base are compatible with or convertible to a format compatible with delivery of information from a global computer network to a general purpose browser, and wherein the concept reader is further executable to create a pointer for each concept that will point to all content associated with each concept.

33. The system of claim 25, wherein the tutor maker program is further executable to:

receive requests from the user over the internet from the user interface;

retrieve concepts within the configuration based on the presentation format; and

send the concepts to the user according to the presentation format.

34. The system of claim 25, wherein the tutor maker program further comprises:

an action handler program executable to receive a request relating to a requested configuration from the user and process the request;

a tutorial manager program executable to:

- determine whether the content of the requested configuration has previously been loaded, and if not, load the requested configuration; and
- determine whether a user session has been established, and if not, establish a user session;

37

a tutorial database containing each loaded configuration; a user session, executable to:

- establish a thread for each user to facilitate synchronization among multiple users; and
- process the request by accessing the loaded requested configuration and generate a response; and

an a format compatible with delivery of information from a global computer network to a general purpose browser generator executable to:

- receive the response from the user session;
- place the response in a format compatible with delivery of information from a global computer network to a general purpose browser format; and
- transmit the response back to the user.

35. The system of claim 34, wherein the tutorial manager is further executable to track a set of referential content for each particular concept knowledge base, problem knowledge base and presentation format knowledge base to ensure that only one instance of each knowledge base is loaded at any one time.

36. The system of claim 34, wherein the tutorial manager program further comprises:

- a concept reader executable to retrieve existing concepts within the concept knowledge base;
- a problem reader to retrieve existing problems within the problem knowledge base; and
- a presentation format reader executable to retrieve an existing presentation format.

37. The system of claim 34, wherein the tutorial database is a collection of pointers that point to a collection of a format compatible with delivery of information from a global computer network to a general purpose browser pages that make up the loaded configuration.

38. The system of claim 34, wherein the action handler processes the request by either forwarding the request to the tutorial manager if a user session has not previously been established, or forwarding the request to the user session if a user session has previously been established.

39. The system of claim 34, wherein the tutor maker program further comprises a presentation format generator executable to generate a user-defined presentation format for the loaded configuration.

40. The system of claim 24, wherein the presentation format generator program is further executable to search a graph representing the concept knowledge base that is based on the hierarchy input by the author using a graph searching algorithm to place concepts within the presentation format based on the order found by the graph searching algorithm.

41. The system of claim 40, wherein the configuration editor program is further executable to allow the author to select a goal concept at least one prerequisite concept, and wherein the presentation format generator program is further executable to initiate the graph searching algorithm using the goal concept as a starting point and the prerequisite concepts as an ending point to collect a subgraph of the concept knowledge base that defines the presentation format.

42. The system of claim 24, wherein the server memory further comprises file memory and random access memory, and wherein the tutor maker program is stored in file memory and the configuration is stored in random access memory.

43. The system of claim 26, wherein the browser extension is further executable to:

- send a request to the tutor maker over the internet; and
- retrieve a response from the tutor maker over the internet, wherein all interactions between the browser extension and the tutor maker are sent through a single URL.

38

44. The system of claim 43, wherein the tutor maker program can determine if the user has sent a request from a different URL than the user had originally set up for the user session and will respond to the user with a query as to whether the user wishes to log off at the original URL location, and if so, logs the user off at the original location.

45. A system comprising a knowledge delivery computer program stored in computer-readable form on a tangible storage medium for delivering concepts from a configuration data base, the knowledge delivery computer program executable by a processor to:

- build the configuration data base to include a plurality of concepts;
- create a configuration hierarchy to define a relationship between each of the plurality of concepts;
- automatically generate a presentation format for the configuration based on the configuration hierarchy; and
- deliver at least a portion of the plurality of concepts to a user according to the presentation format.

46. The system of claim 45, wherein the knowledge delivery program further comprises:

- a configuration editor software program executable to:
 - allow an author to create the configuration by inputting the concepts into the concept knowledge base; and
 - allow the author to build the hierarchy as the author inputs the concepts into the concept knowledge base;
- store the configuration in the memory; and
- a tutor maker software program executable to:
 - allow the user to access the configuration;
 - retrieve concepts within the configuration based on the presentation format; and
 - send the concepts within the presentation format to the user according to the presentation format.

47. The system of claim 45, wherein the concepts input by the author include content, wherein the content can comprise descriptive content, referential content, or both.

48. The system of claim 45, wherein the knowledge delivery program further comprises a user interface software program that the user can use to access the tutor maker program, wherein the user may initiate automatic generation of the presentation format by sending an input to the tutor maker program.

49. The system of claim 48, wherein the user interface further comprises:

- a browser that enables browsing of the network; and
- a browser extension, comprising an event handler executable to allow the user to generate an event to be acted upon by the tutor maker program, wherein the event handler further comprises:
 - a pop-up handler executable to receive an input from the event handler and, if a pop-up window is associated with the input, create the pop-up window for viewing by the user;
 - a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, create the mode change for viewing by the user; and
 - an action sender executable to send a request based on the input from the event handler to the tutor maker.

50. The system of claim 46, wherein the configuration editor program further comprises:

- a concept editor executable to allow the author to add, delete or modify concepts within the concept knowledge base and to create a hierarchy for the concepts in the concept knowledge base, further comprising a concept reader executable to retrieve existing concepts within the concept knowledge base for deletion or modification;

39

- a presentation format generator executable to automatically generate a presentation format for the configuration; and
- a presentation format editor executable to allow the author to manually create a presentation format or to modify an automatically created presentation format within the presentation format knowledge base, further comprising a presentation format reader executable to retrieve an existing presentation format for modification.
51. The system of claim 46, wherein the tutor maker program further comprises:
- an action handler program executable to receive a request relating to a requested configuration from the user and process the request;
 - a tutorial manager program executable to:
 - determine whether the content of the requested configuration has previously been loaded, and if not, load the requested configuration; and
 - determine whether a user session has been established, and if not, establish a user session and wherein the tutorial manager further comprises a concept reader executable to retrieve existing concepts within the concept knowledge base;
 - a tutorial database containing each loaded configuration;
 - a user session, executable to:
 - establish a thread for each user to facilitate synchronization among multiple users; and
 - process the request from the loaded requested configuration and generate a response; and
 - an a format compatible with delivery of information from a global computer network to a general purpose browser generator executable to:
 - receive the response from the user session;
 - place the response in a format compatible with delivery of information from a global computer network to a general purpose browser format; and
 - transmit the response back to the user.
52. The system of claim 46, wherein the knowledge delivery program creates the presentation format automatically by:
- creating a graph representing the concept knowledge base that is based on the hierarchy input by the author; and
 - searching the graph using a graph searching algorithm to place concepts within the presentation format based on the order found by the graph searching algorithm.
53. The system of claim 52, wherein the knowledge delivery program allows the author to select a goal concept and at least one prerequisite concept, and wherein the graph searching algorithm uses the goal concept as a starting point and the prerequisite concepts as an ending point and collects a subgraph of the concept knowledge base graph that defines the presentation format.
54. A computer program product comprising:
- a computer usable medium; and
 - a computer readable program embodied in said medium, said computer readable program operable to deliver concepts to the user, said computer readable program executable to:
 - build a configuration data base to include a plurality of concepts;
 - create a configuration hierarchy to define a relationship between each of the plurality of concepts;
 - automatically generate a presentation format for the configuration based on the configuration hierarchy; and

40

- deliver at least a portion of the plurality of concepts to a user according to the presentation format to inform the user.
55. The system of claim 54, wherein each configuration further comprises:
- a concept knowledge base containing the plurality of concepts;
 - a problem knowledge base containing a plurality of problems relating to the concepts in the concept knowledge base; and
 - a presentation format knowledge base containing at least one presentation format for the concepts in the concept knowledge base.
56. The system of claim 55, wherein the computer program product further comprises:
- a configuration editor software program executable to:
 - allow an author to create the configuration by inputting the concepts into the concept knowledge base and inputting the problems into the problem knowledge base;
 - allow the author to define the relationships among the concepts to build the hierarchy as the author inputs the concepts into the concept knowledge base;
 - store the configuration in the memory;
 - a tutor maker software program executable to:
 - allow the user to access the configuration and to sent an input to the tutor maker program to initiate automatic generation of the presentation format;
 - retrieve concepts and problems within the configuration based on the presentation format; and
 - send the concepts and problems within the presentation format to the user according to the presentation format.
57. The system of claim 56, wherein the computer program product further comprises a user interface software program that the user can use to access the tutor maker program, wherein the user interface further comprises:
- a browser that enables browsing of the network; and
 - a browser extension, comprising an event handler executable to allow the user to generate an event to be acted upon by the tutor maker program.
58. The system of claim 57, wherein the network is the internet and the browser is an internet browser, and further wherein the event handler further comprises:
- a pop-up handler executable to receive an input from the event handler and, if a pop-up window is associated with the input, create the pop-up window for viewing by the user;
 - a mode changer executable to receive an input from the event handler and, if a mode change is associated with the input, create the mode change for viewing by the user; and
 - an action sender executable to send a request based on the input from the event handler to the tutor maker.
59. The system of claim 56, wherein the configuration editor program further comprises:
- a concept editor executable to allow the author to add, delete or modify concepts within the concept knowledge base and to create a hierarchy for the concepts in the concept knowledge base, further comprising a concept reader executable to retrieve existing concepts within the concept knowledge base;
 - a problem editor executable to allow the author to add, delete, or modify problems within the problem knowledge base, further comprising a problem reader to retrieve existing problems within the problem knowledge base;

41

a presentation format generator executable to automatically generate a presentation format for the configuration; and

a presentation format editor executable to allow the author to manually create a presentation format or to modify an automatically created presentation format within the presentation format knowledge base, further comprising a presentation format reader executable to retrieve an existing presentation format for modification.

60. The system of claim 59, wherein the concepts within the concept knowledge base are compatible with or convertible to a format compatible with delivery of information from a global computer network to a general purpose browser, and wherein the concept reader is further executable to create a pointer for each concept that will point to all content associated with that concept.

61. The system of claim 56, wherein the tutor maker program further comprises:

an action handler program executable to receive a request relating to a requested configuration from the user and process the request;

a tutorial manager program executable to:
determine whether the content of the requested configuration has previously been loaded, and if not, load the requested configuration; and
determine whether a user session has been established, and if not, establish a user session;

a tutorial database containing each loaded configuration;

a user session, executable to:

establish a thread for each user to facilitate synchronization among multiple users; and
process the request from the loaded requested configuration and generate a response;

an a format compatible with delivery of information from a global computer network to a general purpose browser generator executable to:

receive the response from the user session;
place the response in a format compatible with delivery of information from a global computer network to a general purpose browser format; and
transmit the response back to the user.

62. The system of claim 61, wherein the tutorial manager program further comprises:

a concept reader executable to retrieve existing concepts within the concept knowledge base;

a problem reader to retrieve existing problems within the problem knowledge base; and

a presentation format reader executable to retrieve an existing presentation format.

63. The system of claim 61, wherein the tutorial database is a collection of pointers that point to a collection of a format compatible with delivery of information from a global computer network to a general purpose browser pages that make up the loaded configuration.

64. The system of claim 61, wherein the action handler processes the request by either forwarding the request to the tutorial manager if a user session has not previously been established, or forwarding the request to the user session if a user session has previously been established.

65. The system of claim 61, wherein the tutor maker further comprises a presentation format generator executable to generate a user-defined presentation format for the loaded configuration.

66. The system of claim 54, wherein the computer program creates the presentation format automatically by:

creating a graph representing the concept knowledge base that is based on the hierarchy input by the author; and

42

searching the graph using a graph searching algorithm to place concepts within the presentation format based on the order found by the graph searching algorithm.

67. A method for delivering data to a user over a network, comprising:

building a configuration data base to include a plurality of concepts;

creating a configuration hierarchy to define a relationship between each concept in the plurality of concepts;

automatically generating a presentation format for the configuration based on the configuration hierarchy; and
delivering at least a portion of the plurality of concepts to the user according to the presentation format.

68. The method of claim 67, wherein creating a configuration further comprises:

creating a concept knowledge base containing the plurality of concepts;

creating a problem knowledge base containing a plurality of problems relating to the concepts in the concept knowledge base; and

creating a presentation format knowledge base containing at least one presentation format for the concepts in the concept knowledge base.

69. The method of claim 68, further comprising:

creating the configuration by inputting the concepts into the concept knowledge base and inputting the problems into the problem knowledge base;

defining the relationships among the concepts to build the hierarchy as the author inputs the concepts into the concept knowledge base;

storing the configuration in the memory;

accessing the configuration;

retrieving concepts and problems within the configuration based on the presentation format; and

sending the concepts and problems within the presentation format to the user according to the presentation format.

70. The method of claim 69, further comprising:

generating a request to receive content from the configuration;

receiving the request relating to a requested configuration;

determining whether the content of the requested configuration has previously been loaded, and if not, loading the requested configuration; and

determining whether a user session has been established, and if not, establishing a user session;

processing the request from the loaded requested configuration;

generating a response; and

transmitting the response.

71. The method of claim 70, further comprising tracking a set of referential content for each particular concept knowledge base, problem knowledge base and presentation format knowledge base to ensure that only one instance of each knowledge base is loaded at any one time.

72. The method of claim 71, further comprising:

receiving the response from the user session; and

placing the response in a format compatible with delivery of information from a global computer network to a general purpose browser format.

73. The method of claim 67, wherein automatically generating the presentation format further comprises:

creating a graph representing the concept knowledge base based on the hierarchy input by the author;

43

searching the graph using a graph searching algorithm;
and

placing the concepts within the presentation format based
on the order found by the graph searching algorithm.

74. The method claim 73, further comprising:

selecting a goal concept and at least one prerequisite
concept;

starting the search of the graph at the goal concept;

ending the search of the graph at the at least one prereq-
uisite concept; and

generating a presentation format including a set of rel-
evant concepts and an order of presentation for the
relevant concepts based on the search.

75. A method for delivering data created by an author at
a network computer connected to a server computer to a user
at a user computer from the server computer over a network,
the method performed by a computer program stored on a
tangible medium, said computer program comprising a
configuration editor program, a tutor maker program and a
user interface program, the method comprising:

executing the configuration editor to:

allow an author to build a configuration data base that
includes a plurality of concepts;

create a configuration hierarchy to define a relationship
between each of the plurality of concepts; and

automatically generate a presentation format for the
configuration based on the configuration hierarchy,
wherein the presentation format comprises a frame-
work for presenting at least a portion of the concepts
and an order in which the at least a portion of the
concepts will be presented;

executing the user interface to:

allow the user to access the tutor maker program and
request delivery of concepts from the configuration;
and

execute the tutor maker program to deliver at least a
portion of the plurality of concepts to the user
according to the presentation format.

76. A system for delivering information to a user of a
computer having a processor and a memory, comprising:

a configuration database stored on the memory and com-
prising a plurality of concepts and including a tax-

44

onomy that defines the relationships between the plu-
rality of concepts;

a knowledge delivery program executable by the proces-
sor to:

create a pedagogy for the configuration; and
deliver at least a portion of the plurality of concepts in
the configuration to the user according to the peda-
gogy.

77. The system of claim 76, wherein the knowledge
delivery program automatically creates the pedagogy for the
configuration based on the taxonomy.

78. The system of claim 76, wherein the pedagogy is
created in response to a user request and based on a set of
inputs from the user.

79. The system of claim 78, wherein the knowledge
delivery program further comprises:

a configuration editor software program executable to:

allow an author to create the concept knowledge base
and the problem knowledge base;

allow the author to build the hierarchy as the author
inputs the concepts into the concept knowledge base;
and

store the configuration in the memory; and

a tutor maker software program executable to:

allow the user to access the configuration;

retrieve concepts and problems within the configura-
tion based on the pedagogy; and

send the concepts and problems within the pedagogy to
the user according to the pedagogy;

and wherein each configuration further comprises:

a concept knowledge base containing the plurality of
concepts;

a problem knowledge base containing a plurality of
problems relating to the concepts in the concept
knowledge base; and

a pedagogy knowledge base containing at least one peda-
gogy for the concepts in the concept knowledge base.

80. The system of claim 78, wherein the set of inputs
comprises the user's familiarity with the plurality of
concepts, a level of detail the user wants, or a format in
which the user wants the at least a portion of the plurality of
concepts presented.

* * * * *

Appendix C: Related Proceedings Appendix

As stated on page 3 of this Appeal Brief, to the knowledge of Appellants' counsel, there are no known appeals, interferences, or judicial proceedings that will directly affect or be directly affected by or have a bearing on the Board's decision regarding this Appeal.